A MODEL OF THE COST OF SOFTWARE DEVELOPMENT FOR THE APOLLO SPACECRAFT COMPUTER

by

DANIEL ALLEN RANKIN

A.B., William Jewell College
(1963)

M.S., University of Washington
(1967)

SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

at the

Massachusetts Institute of Technology

June, 1972

Signature of Author.... Daniel A. Rankin .....
Alfred P. Sloan School of Management, May 12, 1972

Certified by..... Malcolm M. Jones .....
Thesis Supervisor

Abraham J. Siegel

Accepted by.....
Chairman, Departmental Committee on Graduate Students

A MODEL OF THE COST OF SOFTWARE DEVELOPMENT FOR THE APOLLO SPACECRAFT COMPUTER

by

Daniel Allen Rankin

Submitted to the Alfred P. Sloan School of Management on May 12, 1972 in partial fulfillment of the requirements for the degree of Master of Science.

ABSTRACT

The Charles Stark Draper Laboratory of the Massachusetts Institute of Technology has developed and implemented the primary Guidance, Navigation and Control System for the Apollo manned spaceflight project. This thesis examines the software portion of the project to determine a relationship between cost and output of computer programming.

This thesis is primarily a historical case study. The many special conditions of the Apollo software make it difficult to generalize the specific results to other programs. However, the form of the model may be generally useful. Further, case studies of computer programming costs are sufficiently rare that the data should be useful as a reference for managers faced with preparing estimates for software projects.

The methodology employed is quite simple. Accounting data and information on duties of organizational groups are combined to yield costs of various functions in each six month period. Data is available on timing and content of computer programs released. From this data, logic and arithmetic are used to evolve the form and coefficients of the model. The prediction of software cost per period is made on the basis of two measures of output, the total words of coding released in each period and the number of new words of coding in each period.

The following equation closely models the cost of software development in each six month period from 1961 through 1970.

$$\overline{Y}_t = 0.5\overline{Y}_{t-1} + 0.5Y_t = \text{Smoothed predicted cost in period t}$$

Where

$$Y_t = A + C_t + T_t + Comp_t + D_t + M$$

Where

A = Analysis cost = constant cost per period
$$= \frac{(\$39.08/\text{new word})\ (\text{total new words in project})}{(\text{total periods in project})}$$

C_t = Coding cost = ($48.92) (new words released in period t)

T_t = Testing cost = ($13.43) (total words released in period t)

$Comp_t$ = Computer cost = $1.04(A + C_t + T_t)$

D_t = Documentation cost = $0.17(A + C_t + T_t)$

M = Management cost = constant cost per period
$$= \frac{(0.11)\ (\text{total project } A + C + T)}{(\text{total periods in project})} .$$

Thesis Supervisor: Malcolm M. Jones
Title: Assistant Professor of Management

ACKNOWLEDGEMENTS

I would like to express my gratitude to the M.I.T. Charles Stark Draper Laboratory, whose personnel, facilities, and records have made this thesis possible. Grateful acknowledgement is made to my thesis advisor, Professor Malcolm Jones for the kind assistance, helpful insights, and encouragement he provided. To Mr. Robert Millard of the Draper Laboratory go my thanks for his suggestion of the topic and helpful guidance as the ideas for this thesis developed. In addition, Mr. Millard supplied essential data.

I am indebted to Messrs. Kenton Greene, Daniel Dolan and Boyd Watson for their patience in answering my endless questions about the history and organization of the Laboratory and their generousity in making records available. Many other persons throughout the Laboratory compiled data at various times and for various purposes which eventually reached me. These tedious efforts are greatly appreciated, and I alone bear responsibility for the accuracy and use of all data presented herein.

This report was prepared under DSR Project 55-23891, sponsored by the Manned Spacecraft Center of the National Aeronautics and Space Administration through contract NAS 9-4065.

The publication of this report does not constitute approval by the Charles Stark Draper Laboratory or the National Space and Aeronautics Administration of the findings or conclusions contained therein. It is published only for the exchange and stimulation of ideas.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 The Topic of this Thesis

This thesis is a case study of the relationship between computer programming output and cost on a large software project.

The project studied is the software development for the Apollo Primary Guidance, Navigation, and Control system by the Charles Stark Draper Laboratory of MIT. The project output has been the computer programs used on board the Apollo Spacecraft. The cost of this effort, from January 1962 to January 1971, has been about $57 million. The cost to reach the primary objective of the project, the first lunar landing in July 1969 was about $45 million.

The object of this study has been to establish a believable systematic, and reasonably accurate relationship between software output and cost. This has been accomplished by relating the cost of each six month period to two indices of programming activity during that period, total machine words of code released and new words released by spreading and smoothing costs over time.

The model is presented in brief in Chapter II and in detail in Chapter V.

The model is derived from the study of one quite unusual software project. It is unlikely that the exact form and coefficients would apply to any other project. It is hoped, however, that the general form of the model might prove useful for cost estimation or may provide a way to thinking about software development.

1.2 The Nature of Apollo Software

The heart of the on board Apollo Guidance, Navigation, and Control System (GN&CS) is the Apollo Guidance Computer (AGC). The specific aim of the software effort has been to supply carefully verified computer programs and data for the AGC units used on the Command Module and Lunar Module on each Apollo flight. More generally, software includes many activities preceding or supporting computer programming and eventual use of the programs. This includes the theoretical and engineering work which verifies equations and establishes specifications for each phase of the mission. Later chapters, particularly the one

on the duties of the various organizational groups, will give the reader a better understanding of the work which is included as software. Such an understanding is vital to proper interpretation of this paper, for without it the reader may tend to regard the costs as excessive and be unable to make an intelligent adaption of the model to his own situation.

There are several ways in which the Apollo software was probably more difficult or expensive than conventional software projects. Central to several of these points are the problems inherent in a spaceborne computer application.

1) Apollo software includes the derivation, verification, and specification of the concepts, equations, and procedures for the GN&CS functions for all Apollo mission phases.

2) The logical correctness and computational accuracy of the AGC programs must be stringently tested.

3) The software costs include the construction of simulation models of the AGC and spacecraft.

4) The AGC hardware was marginally adequate to meet the demands placed on it, necessitating a high level of optimization of coding.

5) The management/coordination costs were probably

higher than "normal" due to the complexity of the work and the huge size of the multicontractor Apollo project.

6) The personnel on the project, including the programmers, were highly competent, well educated scientists and engineers. This factor raised the average cost of manpower relative to typical projects, but was essential for staisfactory completion of the project.

1.3 The Contributions of this Study

Many authors (Jones and McLean, 1970; Pietransanta, 1970a) have pointed out the difficulty in preparing good advance estimates of software development costs. Realized costs and elapsed time to completion are often well in excess of original estimates, and the preparation of estimates seems to be little more than a black art for experienced managers and outright guessing by inexperienced people.

While the problem of difficulty in estimate preparation is widely recognized and frequently lamented, this writer has found relatively little in the literature of computer science which is of real use to the person who must prepare an estimate. Several

authors have presented outlines of estimating procedures, but without experience and reference historical data it is hard to fill in the steps with hard numbers. The System Development Corporation (SDC) made a survey of 169 computer programming projects in the mid 1960's, in which they attempted to determine the required man months and computer hours as a function of many variables and parameters of the projects. Their study was a useful, pioneering effort. Chapter VI will relate the present effort to the literature.

Given the state of affairs presented above, the first contribution of this paper is simply a detailed prsentation of the costs, outputs and circumstances of a major software project. Despite concern over costs, few organizations have chosen to publish detailed histories of specific software projects.

The second contribution is the model itself, with its use of two output measures and smoothing features to acheive a relationship between cost and programming. This should be useful to persons who must estimate software costs. Such persons should modify the coefficients and perhaps the form to suit their own case, and they should cross check their answers by

other estimating techniques. Admittedly, the problem of estimating the ultimate quantity of new words and total words remains. I hope, however, that the availability of a documented reference case such as this will lend comfort.

A third contribution of this model is the detail it gives on the subdivisions of a software effort. Analysis, coding, testing, management, documentation, and computer costs are all figured separately, then summed to give a total cost for each six month period. Such detail helps illustrate the demands for various types of resources as a function of project completion and in total.

A final use of this model might be project control and programmer monitoring. Such a use would require carefully designed techniques to measure programming output and research to provide cost coefficients appropriate to the project. If even moderately successful, the rewards should be handsome.

CHAPTER II

THE MODEL IN BRIEF

2.1 The Model Equation

The following equation closely models the observed historical cost of the development of the Apollo Guidance Computer (AGC) software. A period is six months.

$\overline{Y}_t$ = Smoothed predicted cost in period t

$= 0.5\ \overline{Y}_{t-1} + 0.5\ Y_t$

Where

Y_t = Unsmoothed predicted cost in period t

$= A + C_t + T_t + Comp_t + D_t + M$

Where

A = Analysis cost = constant cost per period

$= \dfrac{(\$39.08/\text{new word})\ (\text{total new words in project})}{(\text{total periods in project})}$

C_t = Coding cost = \$48.92 x (new words released in period)

T_t = Testing cost = \$13.43 x (total words released in period)

$Comp_t$ = Computer cost = $1.04 \times (A + C_t + T_t)$

Analysis, coding and testing above include all direct and indirect

costs incurred in performing those activities except the cost of the computer facilities used. Computer costs include machine time, operators, supplies, overhead charged to the computer group, and computer group systems programmers.

D_t = Documentation cost = 0.17 x $(A + C_t + T_t)$

M = Management cost = constant cost per period

= 0.11 x (total project A + C + T)/(total periods in project)

2.2 Comparison of Modelled Costs to Actual Costs

Figure 1 presents the cumulative cost of the project as predicted by this model and the actual cost. Figure 2 shows the same comparisons for each six month period throughout the project. The fit is reasonable if one smoothes by eye the erratic changes in the prediction.

The notation I/6X refers to the first half of the calendar year 196X and II/6X to the second half. The data point just above, say, I/6X is the cost for that six month period or the total cumulative cost at the end of that period.

The costs in the model are uncorrected for inflation. They should be regarded as about 1968 dollars.

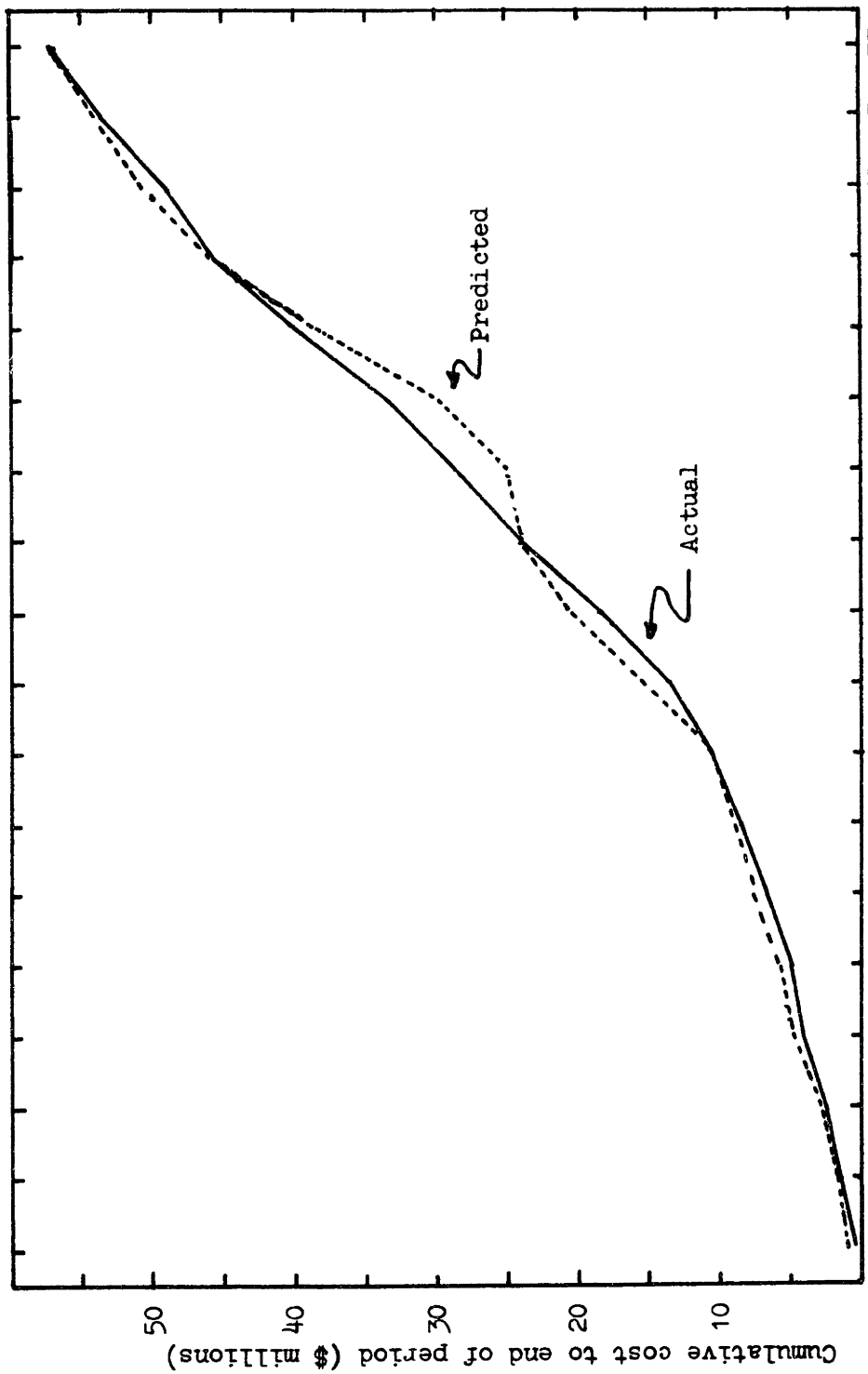

Figure 1. Cumulative total costs, actual and predicted.

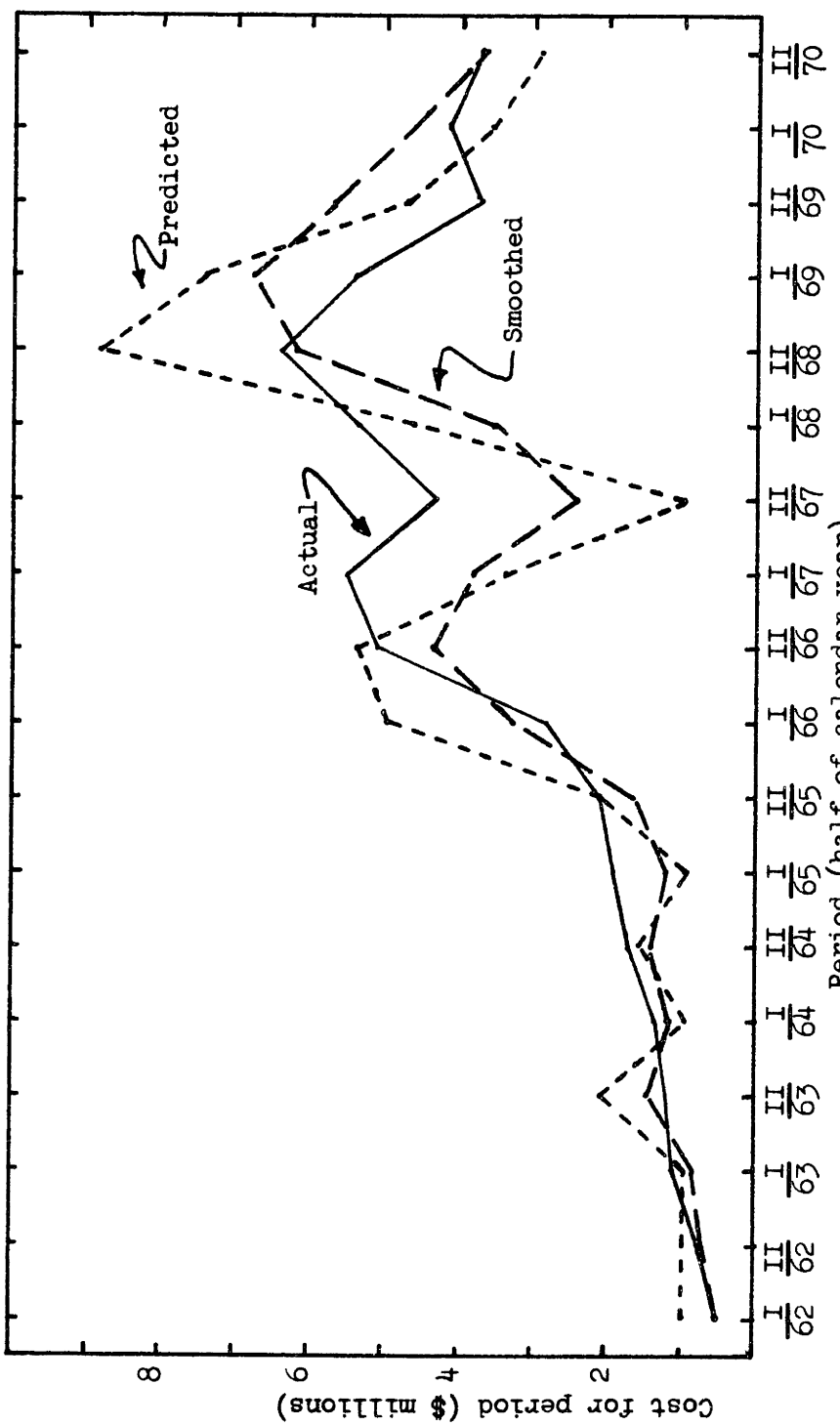

Figure 2. Total cost for each six month period; actual, predicted, and smoothed.

2.3 The Distribution of Costs by Function

Figure 3 shows the distribution of actual costs by function through the first half of 1969. This particular time was chosen because it includes all the effort up to the first lunar landing, which was the objective of the software development. Since that time the project has gone into more of a maintenance and heavy testing mode, complicated by the increased time between Apollo missions.

Referring the Figure 3, analysis, coding, and testing costs (which are defined to exclude computer costs) are all about the same at 15% of the total cost. The computer costs are 45% of the total. If one assumes that computer cost should be split equally between each of the three functions, then each of analysis, coding, and testing with computer time are about 30% of the total with the remaining 10% spent on management and documentation.

Figure 4 shown the percentage distribution predicted by the model equation for a simple project which was completed in one period and had but one release with all new words. The testing fraction of 5.7 percent is probably unnaturally low and should be increased by loosening the definition of "released"

words to include some intermediate stages of development. If computer costs are again assumed equally consumed by analysis, coding, and testing, then we find that analysis costs 32%, coding 36%, and testing 21%, and documentation plus management about 12%.

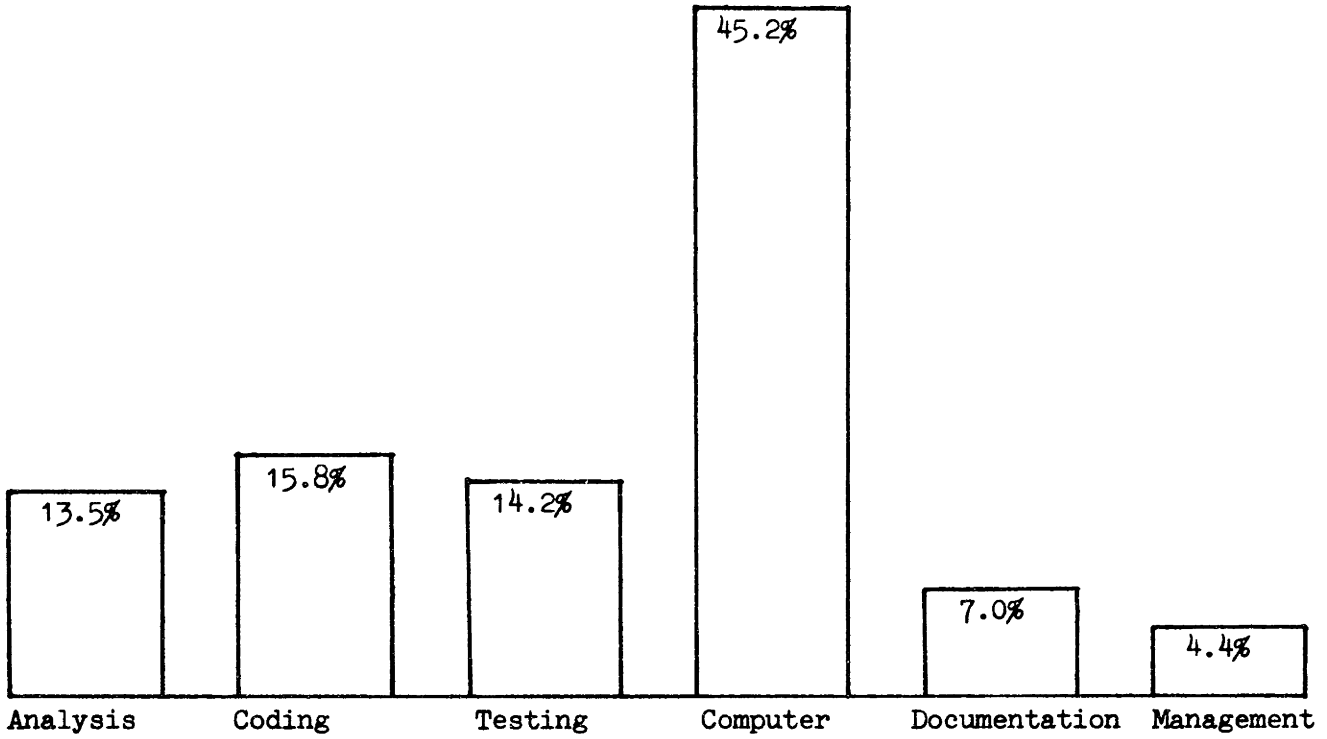

Figure 3. Distribution of effort by function for actual Apollo software expenses through the first complete lunar landing mission.

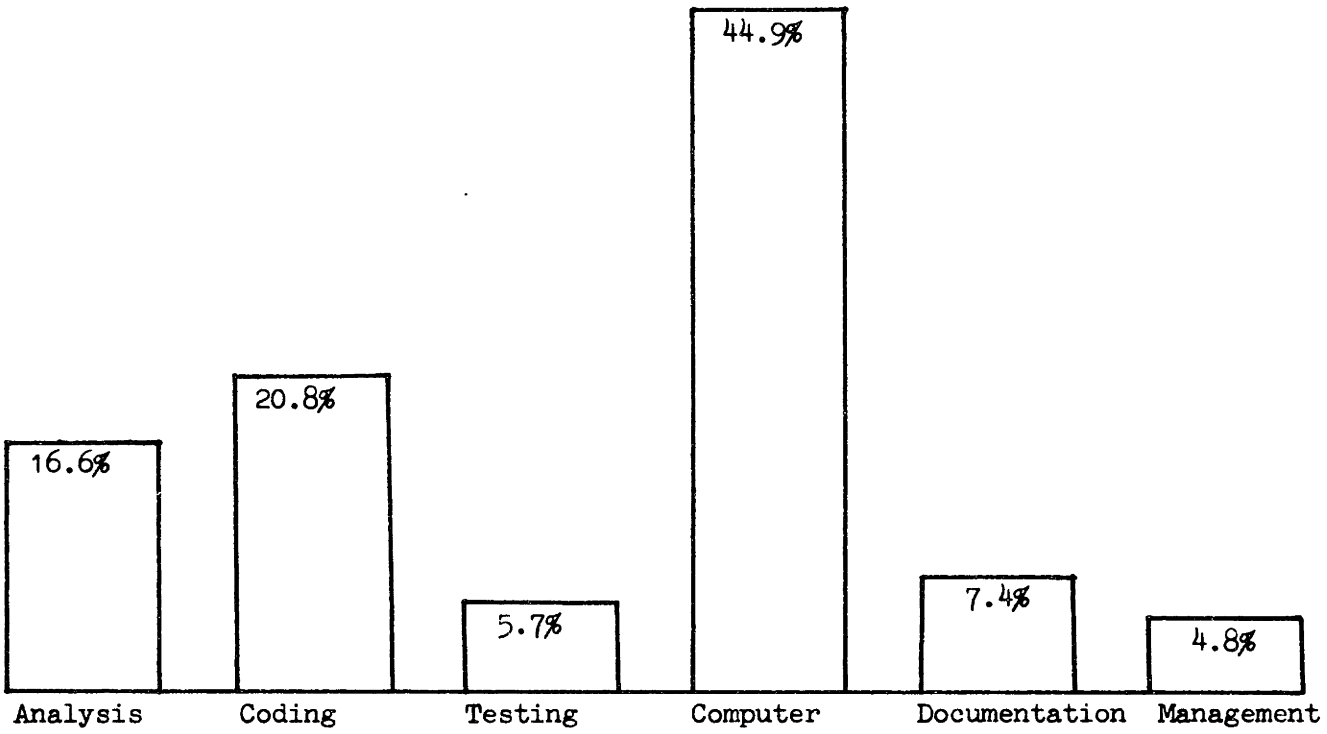

Figure 4. Distribution of effort by function by the model for a hypothetical project completed in one period with one release. (See section 2.3 of the text for a qualifing discussion.)

CHAPTER III

APOLLO GUIDANCE, NAVIGATION, AND CONTROL SYSTEM

3.1 Introduction

This chapter will briefly describe the Guidance, Navigation, and Control System (GN&CS), the Apollo Guidance Computer (AGC) and the computer programs. This material is not completely necessary for an understanding of the model, but is quite interesting and provides relevant background information on the environment.

The material is drawn from three main sources: Johnson and Giller (1971), Mimno (1971) and conversations with Laboratory personnel.

3.2 General

The Charles Stark Draper Laboratory of MIT, formerly known as the Instrumentation Laboratory, has had an important part in the Apollo program. The Laboratory performed a series of conceptual studies of the problems and feasibility of innerplanetary space travel during the late 1950's and gained considerable

experience in aerospace vehicle guidance and navigation from Polaris and other programs. In 1961, when President Kennedy announced the decision that the United States would embark on a manned lunar landing program, the Draper Laboratory received a contract to supply the primary guidance, navigation, and control system.

This assignment involved a substantial amount of work. The various concepts, equation development, and methods of accomplishing the desired ends had to be worked out, or, if known, applied to the project at hand. The on board hardware and software had to be designed, manufactured, and tested. Industrial contractors manufactured the hardware to specifications developed by the Laboratory and approved by NASA. Much co-ordination of effort between the Laboratory and contractors working on other systems was necessary.

3.3 The Apollo Guidance Computer

The Apollo Guidance Computer (AGC) is the center of the on board GN&C system for the Apollo spacecraft. In real time, the computer samples data from and sends commands to

many elements of the GN&C system. Included, as examples, are the Display and Keyboard (better known as the DSKY), inertial navigation equipment, service propulsion engine throttling and gimbals, attitude jets, radar equipment, optical devices, and the like. The computer serves as a data source and receptor for communicating GN&C information to the ground and astronaut crew. It holds a data reservior of spacecraft state vector information and other pertinent parameters about the spacecraft, environment and the mission. All of these functions are controlled and monitored by the astronauts through the DSKY. Finally, there are about 40 functional programs which the crew uses as needed for computations or selects as appropriate for the control of various mission phases.

The computer features a hard-wired fixed memory of 36,864 words and an eraseable memory of 2,048 words. The words contain 15 bits of information and a 16th bit used as a parity check. This total of 38,912 words is the total memory available to programmers. Data can be uplinked from the ground to the computer or entered by the crew, but is limited by the small amount of eraseable storage and to data anticipated by the programs. Reprogramming of the computer on the spot is not possible except

in the limited sense of stringing together existing programs in a useful sequence.

The computer operates in a multiprogrammed mode under the control of an executive program. Normal jobs are processed directly or put into a queue and assigned a priority number. Jobs being executed frequently check the queue to ascertain that no higher priority work is waiting. If such is the case, intermediate results and job reactivation data is stored and attention is focused on the higher priority task. A job may be temporarily suspended, then reactivated if it must wait for I/O operations or otherwise pause.

In addition to the priority system, the computer has two levels of interrupts. The highest is a counter interrupt, used for incrementing the time keeping registers (counts are received and stored at a rate of 100 per minute throughout the mission) and other involuntary, time-important, generally event marking data originating outside the computer. The lower level of interrupts are program interrupts, which start the execution of certain time critical programs.

There are 34 basic machine instructions available to AGC programmers. To expand this number and to provide a library of frequently used routines (such as computation of trigonometric functions and vector manipulations), an interpretive language has been provided. There are many Interpreter instructions, which in effect serve as calls to subroutines which perform the needed calculations.

The Interpreter is somewhat slower than the equivalent machine language steps which it replaces, for added instructions are required for transferring control, location bookkeeping, and the like. Therefore, it is not used in some time critical programs. On the other hand, it saves a great deal of storage by eliminating redundant coding. Storage is very scarce relative to the total demand, and most programs, though not all, have enough slack time to allow use of the Interpreter.

The computer is operated by the crew through the DSKY. This device is shown if figure 5. By means of a succession of keystrokes, the operator calls the program he wishes to use, supplies it with data, initiates execution, and receives or monitors results. One of the special

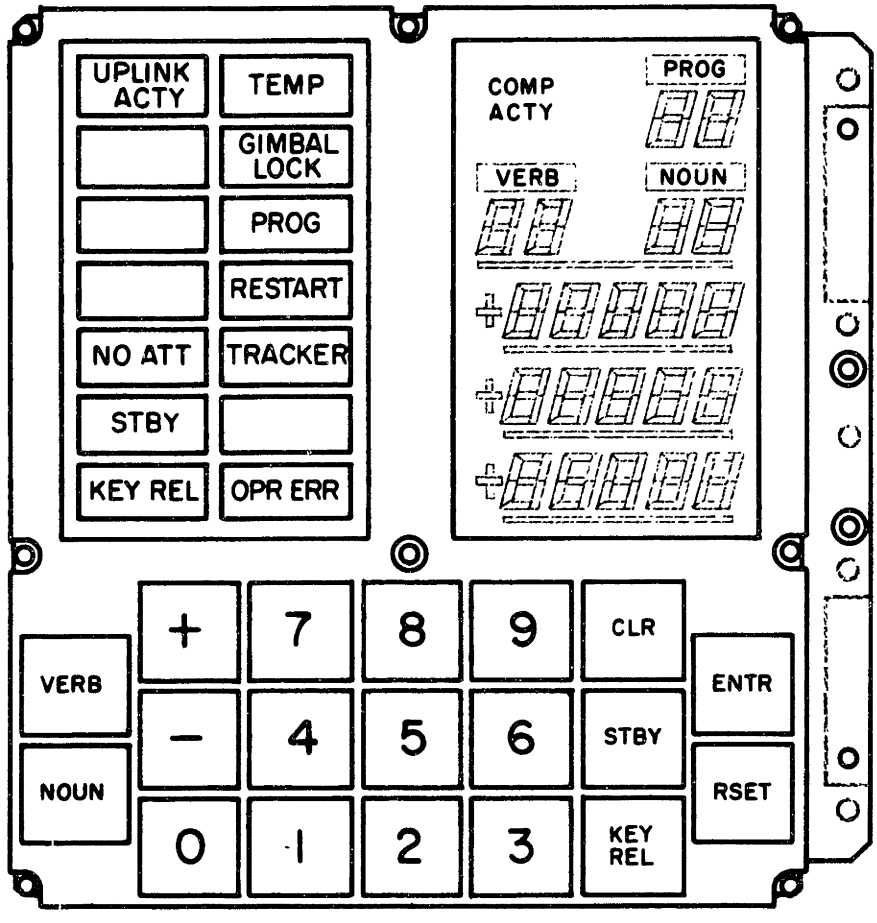

Figure 5. The Display and Keyboard.

characteristics of the computer is its ability to send and receive information via radio link to the ground. It is capable of being fully controlled by ground based personnel.

3.4 Computer Program Evolution

There are two AGC's used on each Apollo flight, one each in the Command Module and the Lunar Module. Each of these machines must have a full set of computer programs for each mission. The programs, as is the case with most other aspects of the two GN&C systems, have much in common but differ in details. Unless otherwise indicated the following applies to either the CM or LM systems.

The entire set of programs and data for a given mission usually totals very near the 38,912 word machine capacity. Such a set of programs is assigned a name and version number and is called an assembly. The assemblies have evolved through time. In the early years of the project they were being created from specifications generated by the analysis groups. Major changes and additions were made as the complexity of the missions increased to full lunar landing capability. In recent years, the

assemblies have been quite stable, with mission to mission changes representing only a small portion of the total coding.

Several months before a flight, the assembly for that mission is formally approved and released. An exact copy of the fixed memory poriton is transmitted to the Raytheon Company where the hard-wired fixed memory is manufactured. This consists, physically, of several boards which are thickly woven with wire. Because of the appearance of these boards, they are informally called "ropes", a term sometimes loosely applied to the assemblies.

Figure 6 (Johnson & Giller, 1971) shows a family tree of the assembly development. Table 1 gives more data on the various assemblies, such as their size and release date. CSM Block II refers to the Command and Service Module Block II GN&C System. This is the system which has been described above and which has been used on all manned Apollo flights. The Block I system featured a smaller but essentially similar version of the AGC. It was used on several unmanned test flights and would have been used on what was intended to be the first manned Apollo flight, AS-204 in early 1967. The tragic Command Module fire occured in the final stages of preparation for this mission.

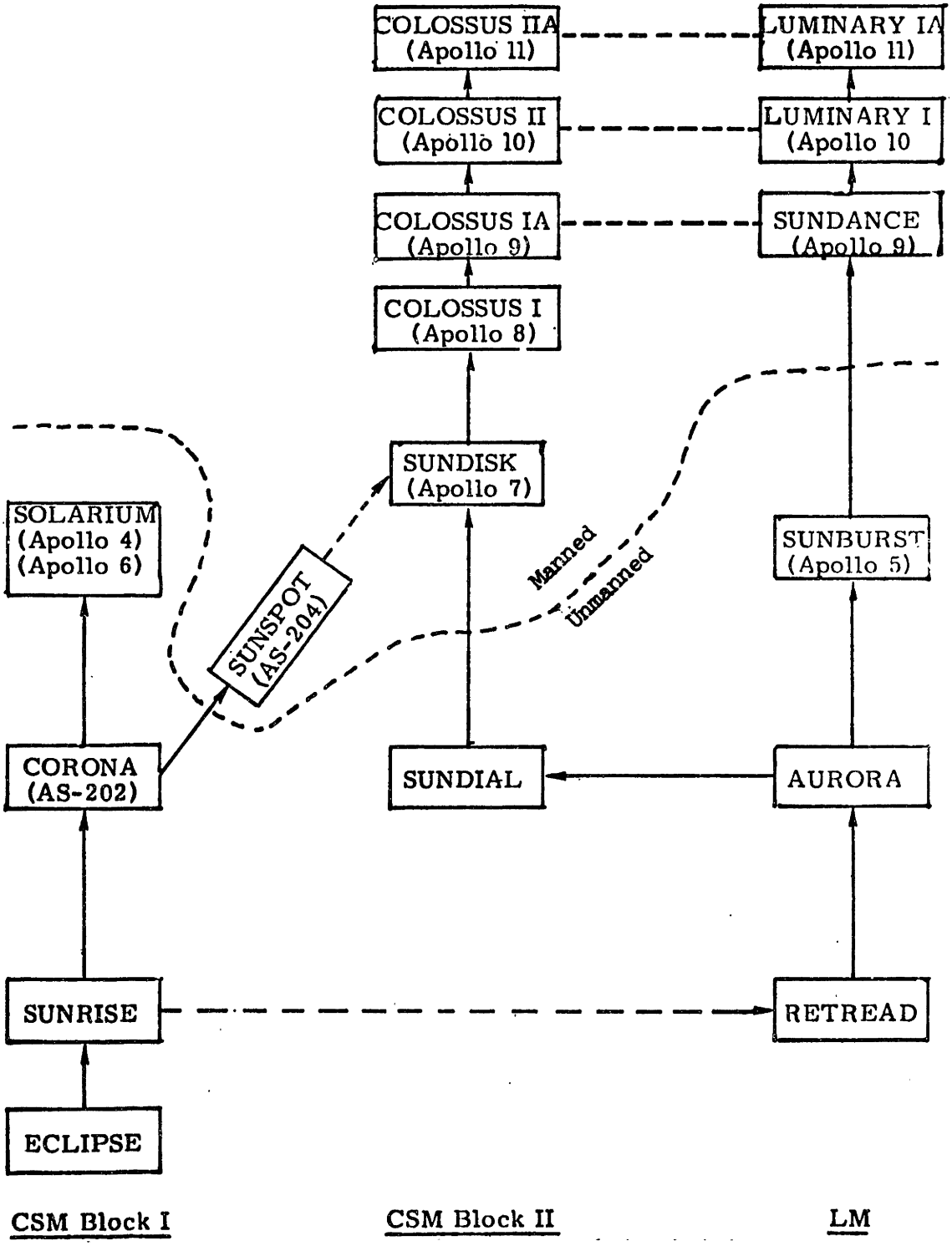

Figure 6. Family tree of assembly development.

TABLE 1 (page 1 of 2). Programming Output of Software Effort.

RELEASE	DATE MONTH/YEAR	APOLLO MISSION	PERIOD HALF/YEAR	NEW WORDS	CUMULATIVE NEW WORDS	TOTAL WORDS	CUMULATIVE TOTAL WORDS
ECLIPSE	12/63		2/63	7978	7978	7978	7978
SUNRISE	10/64		2/64	2750	10728	10451	18429
RETREAD	7/65		2/65	8500	19228	8500	26929
CORONA	1/66	3	1/66	13750	32978	24007	50936
AURORA	3/66		1/66	11000	43978	18297	69232
Summary for period			1/66	24750	43978	42303	69232
SUNSPOT	9/66		2/66	12100	56078	26453	95685
SUNDIAL	10/66		2/66	7700	63778	15685	111370
SOLARIUM	11/66	4,6	2/66	2750	66528	23255	134625
Summary for period			2/66	22550	66528	65393	134625
SUNBURST	3/67	5	1/67	13200	79728	32488	167113
SUNDISK	2/68	7	1/68	23100	102828	38528	205641
COLOSSUS I	8/68	8	2/68	11770	114598	39805	245446
SUNDANCE	10/68	9	2/68	28600	143198	38472	283928

TABLE 1 (page 2 of 2). Programming Output of Software Effort.

RELEASE	DATE MONTH/YEAR	APOLLO MISSION	PERIOD HALF/YEAR	NEW WORDS	CUMULATIVE NEW WORDS	TOTAL WORDS	CUMULATIVE TOTAL WORDS
COLOSSUS 1A	10/68	9	2/68	110	143308	39902	323820
Summary for period			2/68	40480	143308	118179	323820
LUMINARY 1	3/69	10	1/69	10560	153868	39952	363772
COLOSSUS 2	4/69	10	1/69	2035	155903	40623	404395
COLOSSUS 2A	4/69	11	1/69	110	156013	40658	445053
LUMINARY 1A	6/69	11	1/69	2310	158323	40694	485747
Summary for period			1/69	15015	158323	161927	485747
COLOSSUS 2C	7/69	12	2/69	215	158538	40750	526497
LUMINARY 1B	8/69	12	2/69	700	159238	40550	567047
COLOSSUS 2D	12/69	13	2/69	34	159272	40750	607797
Summary for period			2/69	949	159272	122050	607797
LUMINARY 1C	2/70	13	1/70	150	159422	40550	648347
COLOSSUS 2E	5/70	14	1/70	1692	161114	40450	688797
Summary for period			1/70	1842	161114	81000	688797
LUMINARY 1D	9/70	14	2/70	940	162054	40250	729047

Each mission has operational requirements which lead to the specification of software. Previous assemblies are modified and augmented with new programs to meet these requirements, with the individual programs and complete assemblies being throughly tested and verified during development. The assembly is formally approved and released for manufacture several months before the flight, with the interval between release and flight being used for final comprehensive testing. For the purpose of this study, output is considered to have taken place at the time the assembly is released. There were several releases early in the project used for engineering test purposes rather than for a flight. The procedures and testing effort, however, were largely the same and such releases are counted as output.

Table 1 includes data on the size (word count) of the various assemblies. A word is simply the 15 bits plus parity bit which occupy each intentionally used cell of fixed or eraseable memory. Words can be AGC code instructions, Interpreter instructions, or data such as star charts. New words refer to coding or data which has been added or changed from the antecedent release.

The eraseable memory has been counted twice, i.e., as 4,096 words, on some of the later releases. This has been done to better indicate the very intensive useage of these locations. Each eraseable location is used for an average of 6 to 7 different variables during the course of exercising all of the programs in the assembly. This is possible since many of the programs are specific to certain mission phases.

CHAPTER IV

THE PROGRAMMING EFFORT

4.1 Draper Laboratory Organizational Structure and Duties

This chapter will present a picture of the work called software by a description of the organization of the effort. Sections 4.2 and 4.3 will present data on manpower and computer usage.

There has been one major and several minor changes in the organizational structure over the years. The Apollo project was carried on by a small group in the early days, then grew to hundreds of people and became a large share of the total work of the Laboratory at the height of the program. The major change took place in mid 1963, at which time the structure presented below was created. There were a few changes before, and some minor shuffling of groups since, but this description fits well for most of the history.

The Laboratory is divided into groups, each with a number between 00 and 99. There are so-called major groups which have the responsibility for important projects or for the same common

phase of major projects. For example, Group 23, with all its lettered subdivisions, is the main Apollo group, while Group 51, Reliability and Quality Assurance, serves that particular specialized need for Apollo and other projects. In addition, there are groups called minor groups. These groups generally supply services, such as printing.

It is the major groups which perform the direct effort and are of interest here. The following comments on the responsibilities of each group are drawn from a recent contract work statement, organizational charts, and conversations with knowledgeable managers. Appendix A supplements the verbal descriptions with numerical data on the software expenditures and functional duties of each major group. This data was a fundamental input to the model.

Group 23A, the Space Guidance Analysis Division, as its name implies, is essentially an analysis group. Most of its efforts are directed towards "mission oriented" systems analysis and software design. It is responsible for specifying (via systems specifications) the mission systems software with the exception of the Digital Autopilots. Most of its work is carried out using the problem oriented language MAC. In gen-

eral, 23A does not code the AGC, though there are exceptions. Group 23A does not specify the computer operating system for the Apollo Guidance Computer, that work being done by Group 35, Digital Development. Group 23A provides test specifications for proving that the coded guidance computer performs correctly and reviews the test results. The division aids NASA in mission design and proper utilization of the MIT guidance system. Post-flight data is reviewed to assure that the system operated in the intended manner.

While they have not provided the AGC coding, the equation development and specifications which 23A provided were an essential part of software development. All of the work done by 23A has been charged as software.

Group 23B, Mission Program Development Division, has the major responsibility for the actual programming used on board the spacecraft. This involves a great deal of effort, much of which is devoted to what might be called the management and control of coding. The group designs, codes, and documents the programs to the specifications provided by Group 23A and other sources. Extensive testing is required to assure that the coding is correct, both in its logic and in the more

complicated area of the dynamic interactions of the many elements of the GN&C system. A non-trivial example is that the on-off timing of the attitude control jets should not feed energy into spacecraft bending modes. Within the programming itself the number of reasonably conceivable logical paths is effect infinite when one considers all the possible sequences of interrupts and program usage. Tests are carefully designed, run, documented and reviewed. Errors must be corrected, followed by sufficient testing to assure that the fix itself has not introduced further problems. At times, the work and personnel of 23A and 23B are so mingled that the distinctions between them become arbitrary.

Group 23B documents the coding and prepares various user manuals. To maintain careful control over all aspects of the programs, including the interactions with other systems, there are elaborate procedures of program review and approval. These require considerable paperwork and administrative time. The pressure for efficient memory allocation and operating speed/timing constraints requires the establishment of committees which coordinate and often pass judgements in this area. During missions, members of this group take an active

part in monitoring the flight and if needed are available to supply information or services. Such contributions have been required on several missions. Post-flight reviews are made to ascertain actual program performance. A number of service tasks are performed by this group to translate the efforts of the programmers at their desks into test runs, update master programs, and ultimately put the final programs into the memory banks of the flight aritcle computer.

All of the expenses of this group are considered part of the software effort.

Group 23C, Control and Flight Dynamics Division, is primarily concerned with the design development, and support for the Digital Autopilot. The Digital Autopilot, which runs on the AGC, performs the stabilization and control functions. The logical design and coding of the programs involved is among the most complicated of the entire software package.

All of the work of Group 23C is considered software.

Group 23D, Display and Human Factors Division, has its primary duties in the areas of mission verification, design support of proposed software changes, and procedural reviews.

Additional areas include human factors design studies and crew training. One of the important functions of this group was the construction and continuing maintenance of mockups of the Command Module and Lunar Module, which are connected to simulated guidance, navigation, and control systems. The cockpits provide operating hardware in a realistic environment. They are tied to the Hybrid Simulator, which is a system of analog and digital computers much used in testing of flight computer programming.

The activities of Group 23D have supported both hardware and software development. In the early years of the program, this group was designing and testing the astronaut interface equipment and mission procedures, plus the design and construction of the module mockups. As the program has matured their duties have changed almost completely to software support. The proportion of their expenses chargable to each has been estimated for each time period and the software portion considered in the software cost.

Group 23H, Hybrid Computing Division (earlier known as Group 25) has charge of the creation, maintenance, and use of the Hybrid Simulator. The Hybrid Simulator, mentioned in

the discussion of Group 23D, combines analog and digital computers with appropriate pieces of spacecraft hardware (notably an AGC computer and its human I/O device, the Display and Keyboard) to provide a real-time simulation of mission operations using real AGC software. Simulations can be run with or without use of the module mockups, depending on the purpose of the run. This simulator can provide powerful tests of software and proposed mission operating procedures. The analog computing components are capable of providing high speed response to spacecraft dynamics and mechanical feedbacks, things which require much time to simulate on digital machines. The digital computers in the system control the tests, do the part of the computing which they perform well, and provide extensive data collection during simulations for later analysis. The use of real time simulation and spacecraft hardware provide information on software performance in as real an environment as possible and very important data on human response and capabilities relative to the demands of the astronaut interface equipment.

Group 23H is charged wholly to software.

Group 23P is entitled Project Management. It includes the senior line managers of the Laboratory's Apollo effort and the administrative staff. The staff serves two major functions. One is handling the matters related to contracts, subcontracts, budgets and reports. The other is structuring, assisting in, and maintaining the procedures which plan, control, and coordinate the work on Apollo. Such control systems exist both internally and with NASA and other outside groups. Group 23P does not do all the work associated with project control, but does oversee the system and serve as a point of contact with NASA and other contractors.

The expenses of Group 23P have been split between hardware and software roughly in proportion to the total hardware/software split of the operating groups. To the end of 1964, one third of 23P is considered software, since then, one half.

Group 23S, Systems Engineering Divison, is a relatively small group. It designed and developed the overall software specifications for NASA and performed mission planning and support, primarily with and in direct support of NASA. This group does system engineering to assure that systems are

compatible, meet specifications, and are logically complete. They assure that the efforts and outputs of the Laboratory are complete and consistant. Their work is considered all software, expect for one third hardware early in the program.

Group 23T, System Test Group, runs a facility which includes a complete, operating, Guidance and Navigation System for both the Command Module and the Lunar Module. These Engineering Test Simulators are used to test the interface between real hardware and software, as well as development and tests of the hardware itself. Most of the efforts of this group over the years has been hardware related, but a small percentage, ranging from 10 to 34 percent, has been allocated to software costs to cover testing of programs on hardware.

The last major group which contributed to software is the Digital Computation Group, Group 33. They supply and operate the digital computation facilities of the Laboratory. During the height of the Apollo programming and testing effort, for 18 months in 1968-69, two IBM 360/75 computers were running around the clock. Each had a large core memory and extensive peripheral devices. One of these machines remains in place today. Group 33 operated these machines, supplied

the systems programmers who set up and maintained the operating systems, and constructed the digital simulators of the AGC and Apollo spacecraft. Section 4.3 discusses computer usage in greater detail.

The digital simulator has been a very important and heavily used testing device for AGC coding. It has two major components. The first is an AGC instruction simulator which precisely simulates the characteristics of the AGC on the big computer. The second component is an environmental simulator which responds to the outputs of, and supplies inputs to, the simulated AGC computer as would happen for a real AGC in a spacecraft during actual mission operations. It is this simulator which the AGC programmers use to develop and test their coding. Many forms of diagnostics and traces for tests are available. The Hybrid Simulator and the Engineering Test Simulator provide useful tests on finished programs and confirm various software/hardware/human interfaces. The Digital Simulator provides a program development facility and powerful tests of completed programs.

Group 33 distributes its costs to projects in accordance

with usage of computer time by programmers working on those projects. All of the computer time charged to Apollo has been considered as used in the development of software.

The minor groups and allocated charges in the Laboratory are charged to various major groups in proportion to the salaries, wages and overhead expended by the major groups. Thus the total expenses of the Apollo effort of the Laboratory are contained in the calculated total expenses of the mjaor groups which have worked on the project. From the knowledge of the hardware/ software effort split of major groups, a full cost for software is obtained.

This discussion has not detailed the groups which devoted their efforts to hardware. This includes the development of the computer, the inertial navigation equipment, radar systems, and various interface hardware. Roughly speaking, over the life of the project up to the first lunar landing, about two thirds of the expense was for hardware and one third for software. All of the actual manufacturing of hardware components was done elsewhere.

The AGC was developed by Group 35, Digital Development. This group designed the processor architecture and the

operating system for the AGC. This important, well done work was performed by a small number of very sharp persons early in the program. However, no part of the Group 35 effort has been allocated to software. This bit of programming, however vital, was but a small fraction of their work, had a relatively low dollar cost, and has been treated as computer development (i.e., hardware) rather than mission software.

The dividing line between hardware groups and software groups, and between such duties within groups is ambiguous. It is difficult to make precise classifications of the two types fo work, and even if one could, the records from which to produce exact dollar costs are not available. In any case, the substance of the model does not require this great a precision in the data. It is much more important that the reader understand the definitions used for the somewhat subjective cost separation than worry about marginal judgements in the application of the definitions. The description of group functions is intended to facilitate such understanding.

4.2 Manpower

The data for this section and section 4.3 were obtained from data published by Madeline Johnson and Donald Giller (1971).

The software effort staarted slowly quite early in the project. The design work was done in the early years by a few people and comprised only a small fraction of the cumulative effort, but was absolutely critical in that it laid a successful framework for the vast work performed later. Figure 7 shows the number of personnel assigned to Apollo for hardware and software.

Some programmers were, and still are, subcontracted personnel. This was done to provide enough skilled people at a time when they were very hard to find and to minimize layoffs of recently hired "permanent" personnel after completion of peak requirements. These programmers worked in offices in the Draper Laboratories, and have become indistinguishable from regular employees except on paper.

"Programmers" and "engineers" have been largely one and the same in preparation of software. Particularly in the early years, the persons who established techniques and spe-

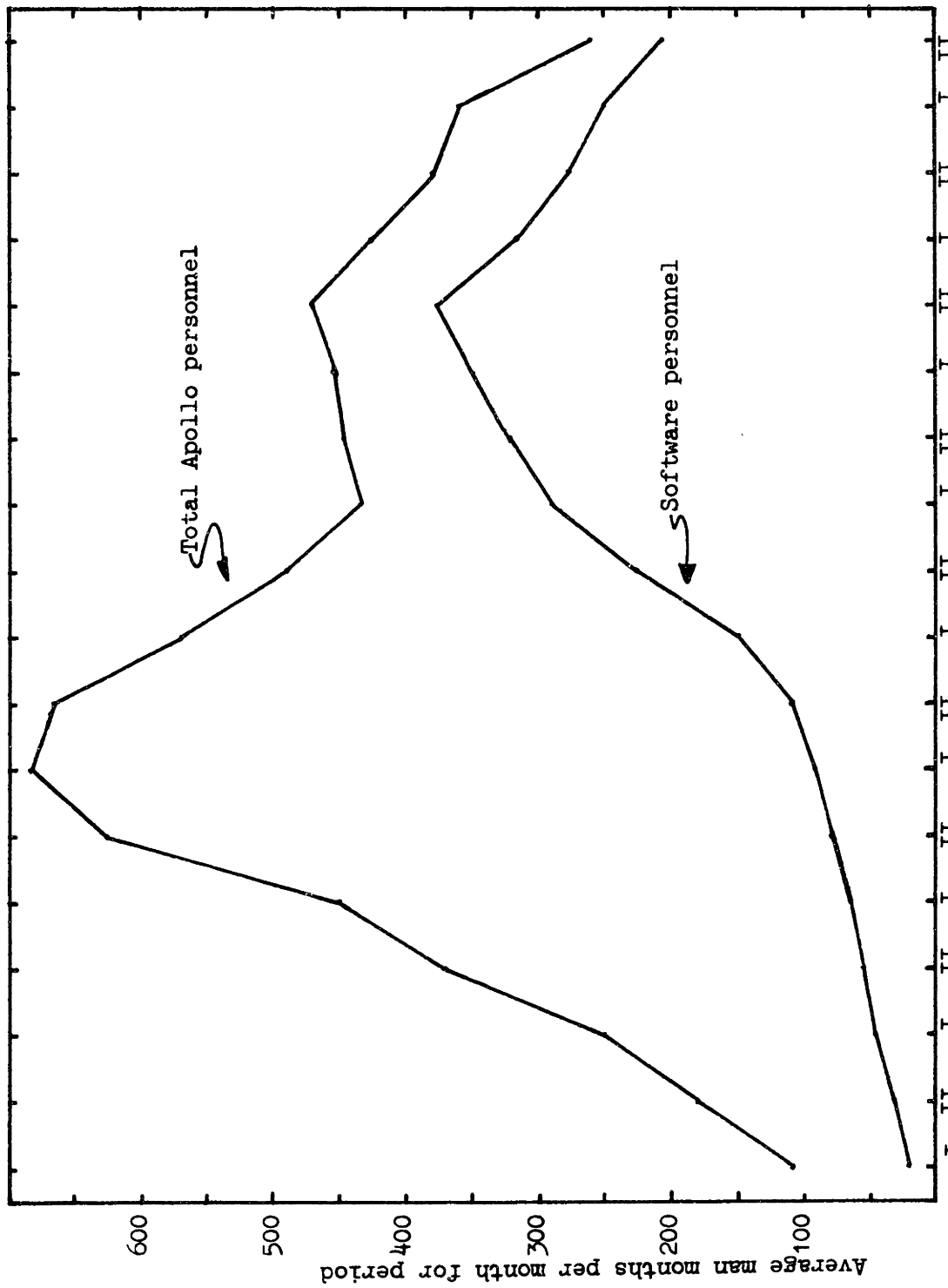

Figure 7. Manpower usage by the Draper Laboratory Apollo effort.

cifications for the GN&C systems also coded and tested AGC programs. As time passed, the increasing size of the project forced greater specialization and formalization of duties. In many cases, specific mission phases would be handled by a small group of persons which included analysts (Group 23A) and computer programmers (Group 23B). These task assignments would link people across organizational group boundaries. While the analysts and mission designers slowly moved away from direct AGC coding and testing, the programmers by and large continued to be highly competent engineers trained to specialize in AGC programming.

4.3 Computer Usage

Figure 8 (Johnson & Giller, 1971) shows a record of the digital computation facilities and usage under the direction of Group 33 from 1959 through 1970. The vertical scale is logarithmic, with unity representing one equivalent Honeywell H1800 CPU hour per month. This graph shows the type of computer equipment in use at various times. The mainframe units were usually configured in nearly their maximum core memory size and with a host of peripheral equipment. By the time they were

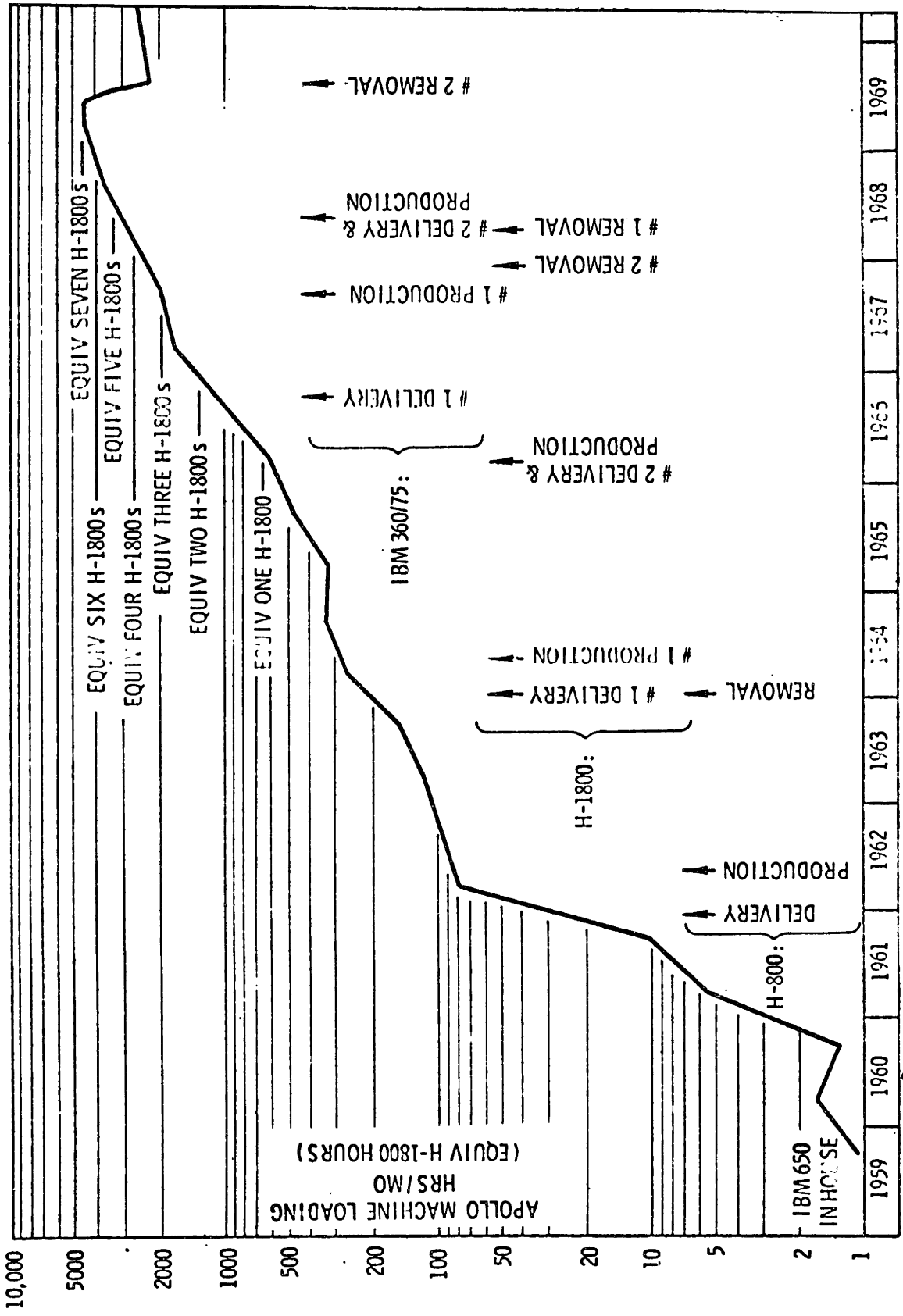

Figure 8. Computer usage by the Draper Laboratory Apollo effort.

replaced, each of the three basic systems (H800, H1800 and IBM 360/75) had been stretched to the maximum feasible size. Even so, there has rarely been a surplus of computing time available. The notation "in production" means that the AGC simulator was in full operational order. This usually happened sometime after the equipment was routinely available for normal computation.

For the purpose of making this figure, it has been estimated that the IBM 360/75 is four times as powerful as the Honeywell H1800 and that the Honeywell H800 provides approximately one third the power of the H1800.

CHAPTER V

THE MODEL IN DETAIL

5.1 Introduction

This chapter will examine in detail each term of the model and show how it relates input and output.

The data presented in this paper came from a variety of sources. The administration personnel for Group 23P have processed accounting records into information on the costs of each group. Mr. Robert Millard has quantified the functional duties of organizational groups. Various persons throughout the Laboratory have compiled data on the content of released AGC computer programs. The contribution of the author has been the design of the model and some collection and reworking of data into appropriate forms as the model has developed. Specific references are not given, for the data simply comprise the records of the organization. The author alone is responsible for the use or misuse and accuracy of the data and conclusions presented herein. His sincere appreciation goes to those who have provided the summaries and other information which made this study possible.

5.2 Data on Cost Input and Programming Output

The model (please see page 17) was constructed by a search for a general relationship between cost and computer programming output. The various data were assembled, after which logic and simple arithmetic were used to evolve the form and numerical coefficients.

The computer programming output is shown in Table 1. The two measures which have been found most useful in establishing correlations between cost and output are the number of new words released and the total number of words released. As described in section 3.3, a release is the formal event in which the programs for an AGC unit for a specific flight are passed from the Laboratory to NASA. New words are coding which represents new or substantially rewritten capabilities since earlier releases. Total words means the total number of words in the release, the sum of new words and coding carried forward from previous releases. These two measures are used as a series of six month totals.

The basic input to programming is the dollars spent on software. This total cost has been divided into six functional areas:

Analysis (A), Coding (C), Testing (T), Computer costs (Comp), Documentation (D), and Management (M). Table 2 shows these costs for each six month period.

The data for table 2 come from the multiplication and summing of two sets of data. The first is accounting records which give the expenditures of each of the organizational groups. The second set of data is a distribution of the work functions of each group into first, hardware and software; and secondly, fractions of the software component expended in each of the functional areas.

The accounting records have been processed in such a manner that the sum of the cost of the major Apollo groups is the total of the Laboratory expenses for Apollo work. All overheads, indirect expenses, minor groups, and the like have been allocated to the major groups. Thus the figures for software are complete and represent the full cost. A breakdown of the cost figures for each group are given in the Appendix.

Section 4.1 of this thesis described the duties of each group. Mr. Robert Millard, head of Group 23P, Apollo Management, used his long association with the project as the basis for

TABLE 2 (Page 1 of 2). Period and Cumulative Costs. (Thousands of Dollars)

PERIOD (HALF/YEAR)	1/62	2/62	1/63	2/63	1/64	2/64	1/65	2/65	1/66
Analysis	223	319	460	493	490	432	383	339	444
Cum. Analysis	223	542	1001	1494	1984	2416	2800	3139	3583
Coding	39	106	153	183	167	211	251	347	505
Cum. Coding	39	146	299	482	649	860	1112	1459	1964
Testing						25	66	124	238
Cum. Testing						25	92	216	454
Digital Computer	180	278	447	360	461	449	450	653	1008
Cum. Digital Comp.	180	459	906	1266	1728	2176	2627	3280	4288
Hybrid Computer	79	43	37	38	59	270	406	302	232
Cum Hybrid Comp.	79	122	160	198	256	526	932	1234	1466
Documentation				74	71	109	149	157	143
Cum. Documentation				74	145	254	403	561	704
Management				71	100	173	181	178	196
Cum. Management				71	171	344	526	704	900
Total	522	746	1097	1219	1348	1670	1888	2100	2767
Cumulative Total	522	1268	2366	3585	4933	6603	8491	10591	13358

TABLE 2 (Page 2 of 2). Period and Cumulative Costs. (Thousands of Dollars)

PERIOD (HALF/YEAR)	2/66	1/67	2/67	1/68	2/68	1/69	2/69	1/70	2/70
Analysis	535	459	368	412	438	324	216	147	98
Cum. Analysis	4118	4577	4944	5356	5794	6118	6334	6481	6579
Coding	631	1034	731	996	1054	748	302	403	372
Cum. Coding	2594	3628	4359	5355	6409	7157	7459	7862	8234
Testing	583	891	651	1031	1498	1339	1094	1139	1107
Cum. Testing	1036	1927	2578	3610	5107	6446	7541	8680	9787
Digital Computer	1416	1648	1506	1434	2446	1964	1020	1365	1288
Cum. Digital Comp.	5704	7352	8858	10293	12739	14703	15722	17087	18375
Hybrid Computer	1391	844	544	816	334	442	577	515	261
Cum. Hybrid Comp.	2857	3701	4245	5061	5395	5838	6415	6930	7191
Documentation	280	468	355	482	455	419	339	374	302
Cum. Documentation	984	1452	1807	2289	2744	3164	3503	3876	4179
Management	258	141	181	183	164	181	189	221	237
Cum. Management	1158	1299	1480	1663	1827	2008	2197	2418	2655
Total	5094	5485	4336	5355	6388	5418	3737	4165	3665
Cumulative Total	18452	23937	28273	33628	40016	45434	49170	53335	57001

a quantitative distribution of group effort into specific functions. This was done before the relationships in the model equation were discovered.

The distribution of costs between hardware and software and among functions in software depends to some extent on definitions and judgement. The length of exposition on group activities and software functions is in part an effort to give the reader an operational understanding of what has been done. The figures in Appendix A reflect various adjustments for extra-ordinary accounting entries which do not properly reflect actual period software efforts.

5.3 The Terms in the Model

5.3.1 Analysis cost

Figure 9 shows the cost of analysis for each six month period and cumulatively through the effort. Reasonably enough, analysis cost begins early in the program. In this particular experience, the absolute cost remained approximately steady until a decline late in the program. Early, the issues are still general and the relatively few persons working on the project spent

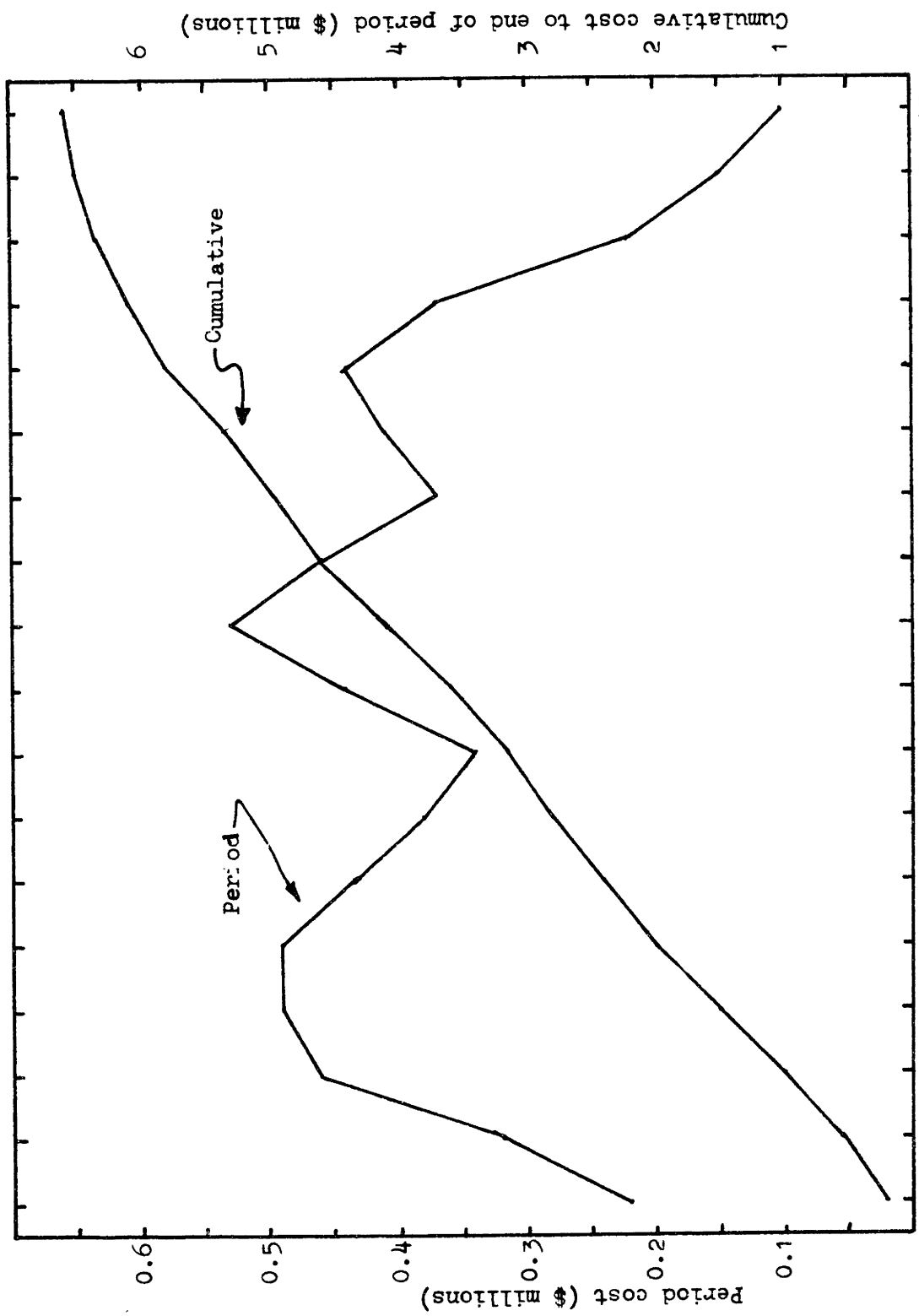

Figure 9. Period and cumulative analysis cost. (exclusive of computer cost)

most of their time on analysis to lay out the grand design. But this does not end analysis, for ever more detailed levels of specification are necessary. Analysis becomes a smaller portion of the total cost as coding and testing grow, but remains important on an absolute cost basis. Some analysis must continue until the last flight is made.

Analysis cost has been assumed to be proportional to the number of new words of code introduced. Figure 10 shows period and cumulative plots of new words against time and Figure 11 shows cumulative analysis cost against cumulative new words released. The fit is clearly not directly proportional in the short run, for analysis cost is heavily front-loaded. Problems are attacked before, perhaps even by several yeras, the time that the solutions to them appear in official releases.

The model assumes that the total analysis cost is fixed by the total number of new words eventually coded, while the distribution of this expense is even through time. The cost per word for analysis in this experience has been $39.08, exclusive of computer cost. The expression shown in the model equation is designed to spread this cost evenly over the life of the project.

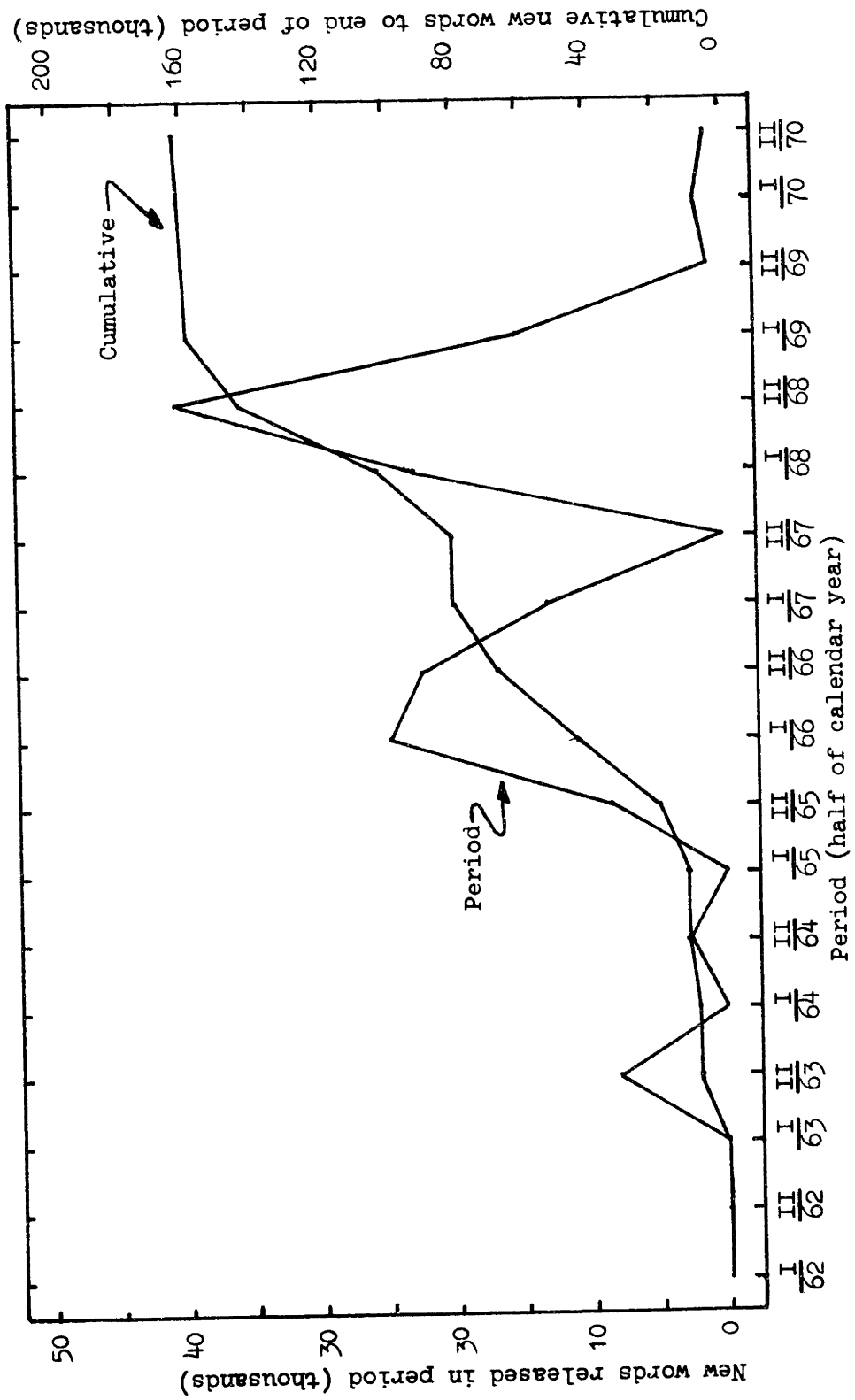

Figure 10. New words in flight releases, by period and cumulatively.

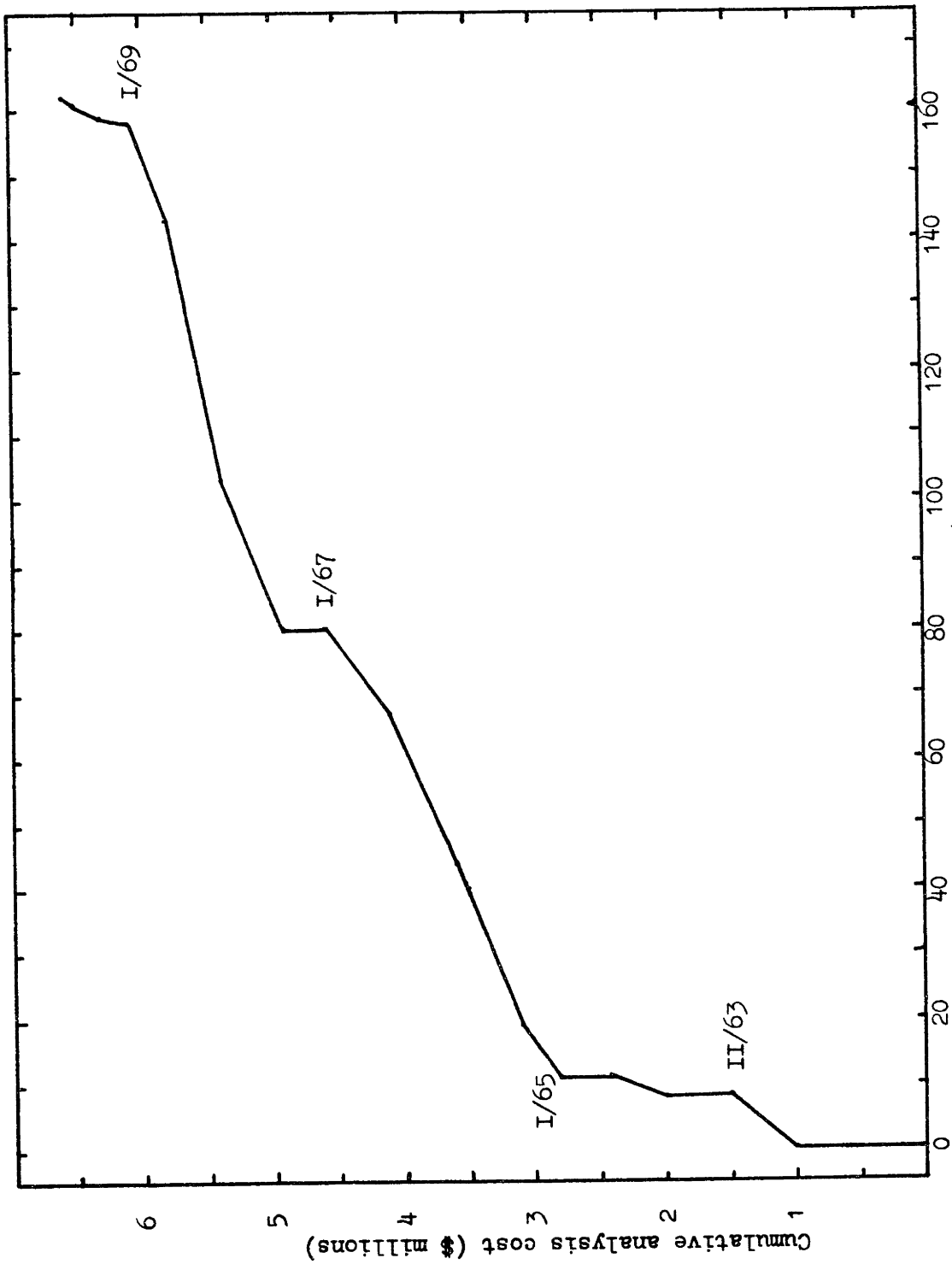

Figure 11. Cumulative analysis cost vs cumulative new words.

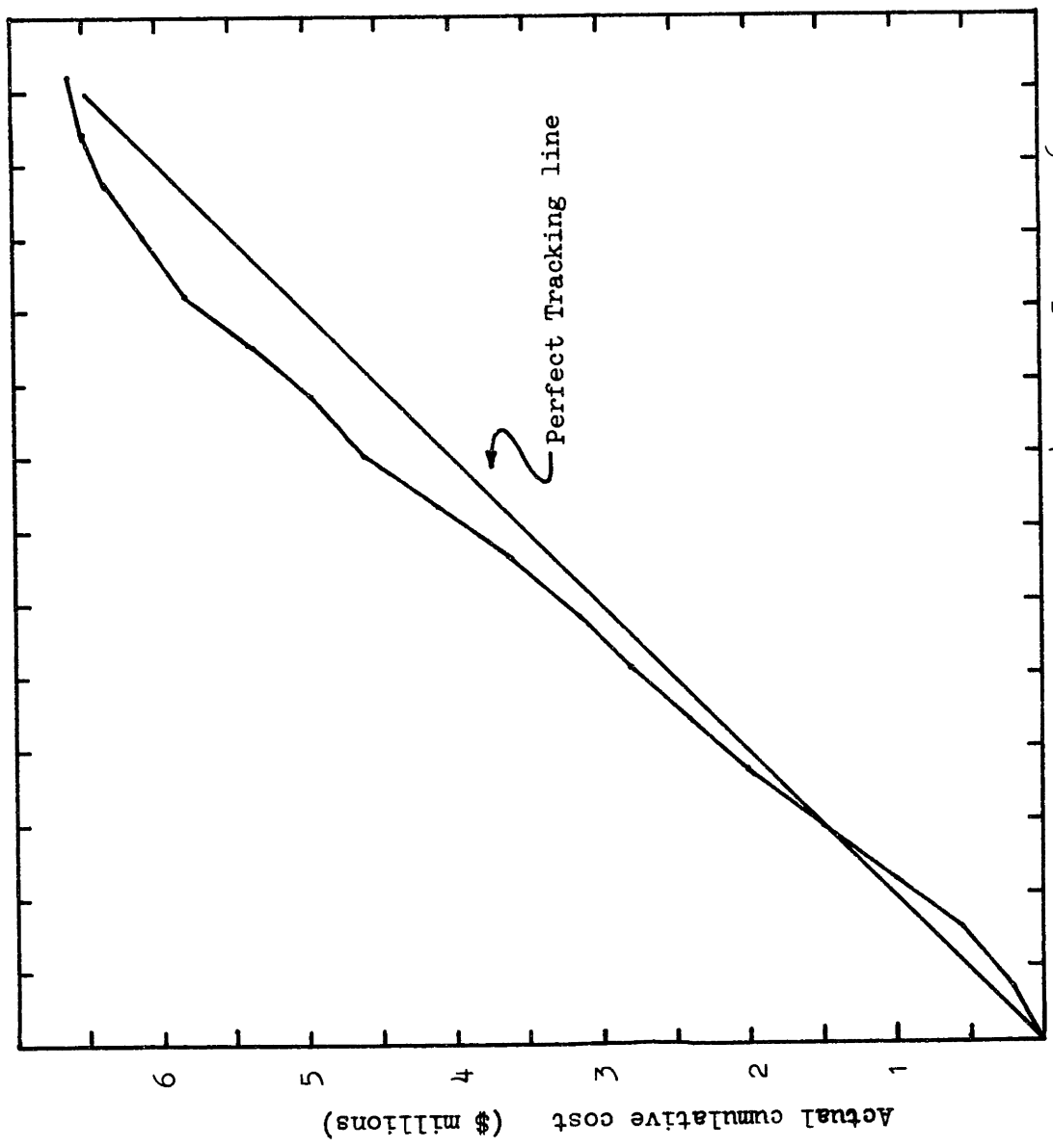

Figure 12. Actual vs model analysis cost.

Figure 12 shows the actual comulative analysis cost vs. model cumulative analysis cost.

5.3.2 Coding cost

The cost of coding, exclusive of computer usage, in the model is taken as proportional to the number of new words released. No time delay is used, and while there is obviously a delay in real life, the use of six month aggregate time periods mitigates the problem. In any case, Figure 10, new words released, Figure 13, coding cost, and Figure 14, cumulative new words against cumulative coding cost, show that a constant cost per new word provides a reasonably valid total coding cost figure throughout the life of the project. In this experience, the cost has been found to be $48.92 per new word.

Coding is defined to include the testing involved in debugging small program modules and the basic integration of these modules into larger sections of the assemblies. Testing, the next cost category, refers primarily to large scale integration, comprehensive testing, and verification of the assemblies or major sections thereof.

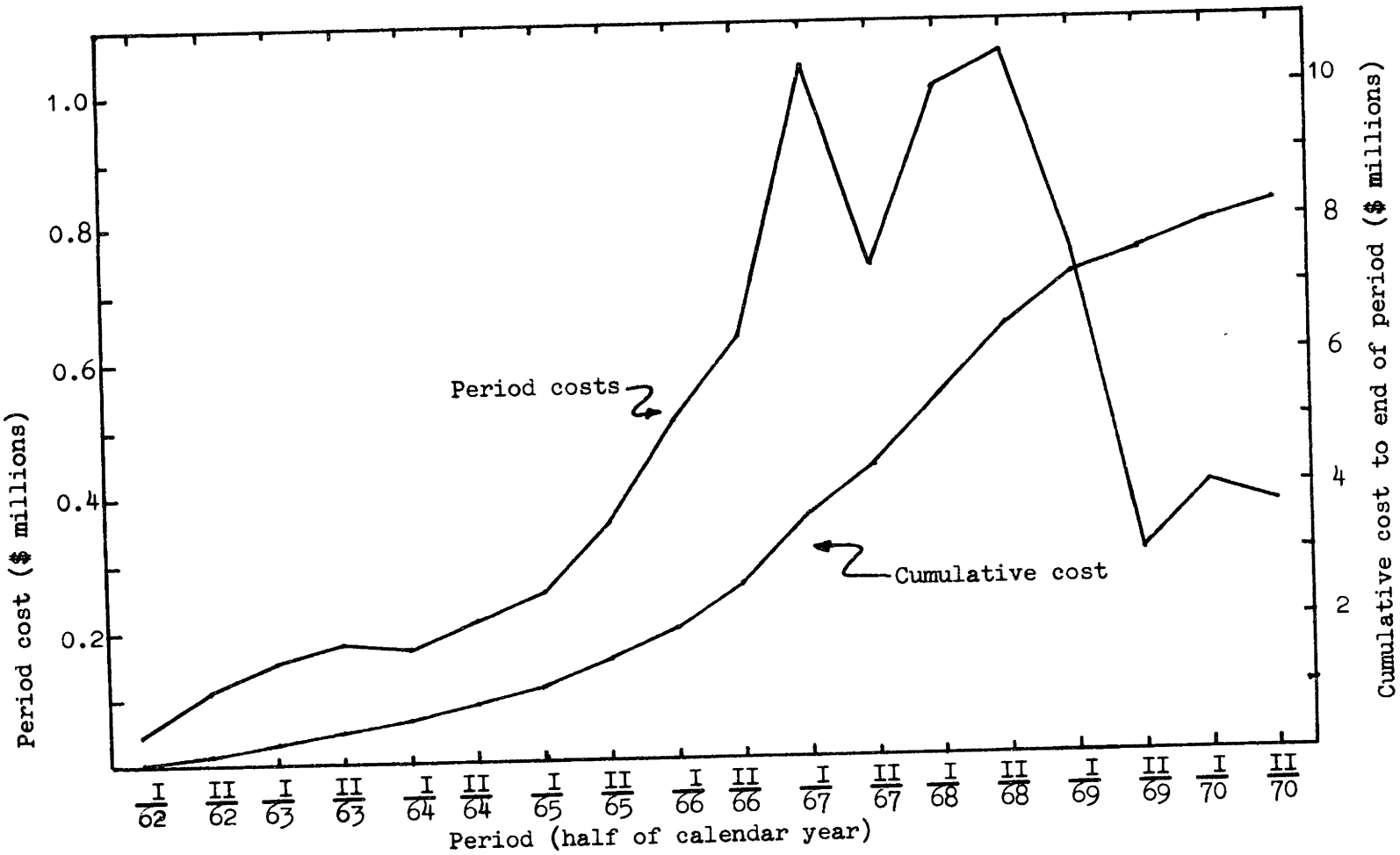

Figure 13. Period and cumulative coding cost. (Exclusive of computer cost)

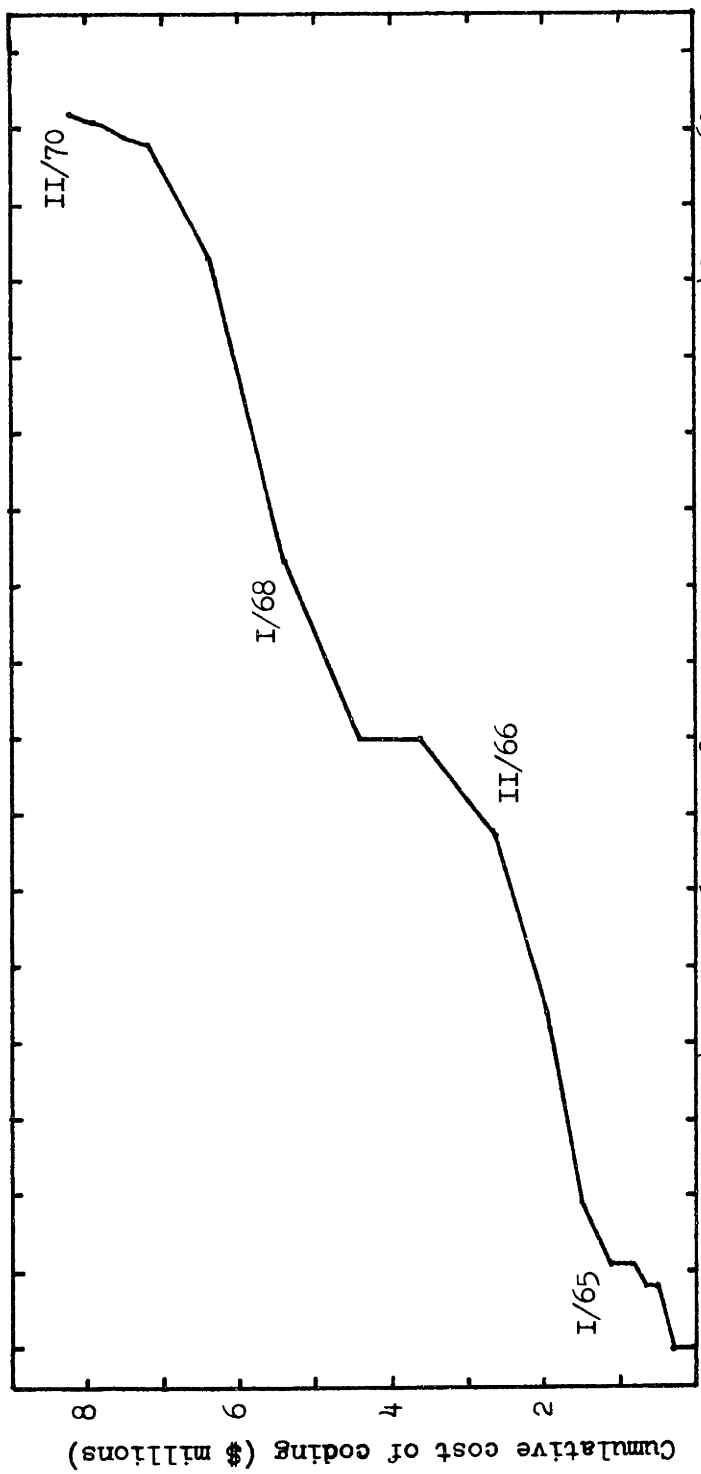

Figure 14. Cumulative coding cost vs cumulative new words. (Excluding computer cost)

5.3.3 Testing cost

Figure 15 shows the cost of testing through the life of the project, again exclusive of computer time. Figure 16 shows the total number of words in flight releases. Figure 17 compares cumulative testing cost and cumulative total words released, demonstrating a steady linear relationship between them. This figure leads to the conclusion that testing cost is about $13.43 per word flight released.

This conclusion is reasonable when one considers the activity of the Laboratory. Even though only a fraction of the words in an assembly are new at the time of a given release, the entire assembly must be throughly tested before men trust their lives to it. The coding is complicated due to the great pressure to conserve all possible memory space and execution time. The eraseable memory locations are used for an average of 6 to 7 different variables in the total collection of programs which make up an assembly. It is a major task to assure that no unforeseen conflicts arise from such intensive usage. Even minor changes in coding dictate comprehensive retesting.

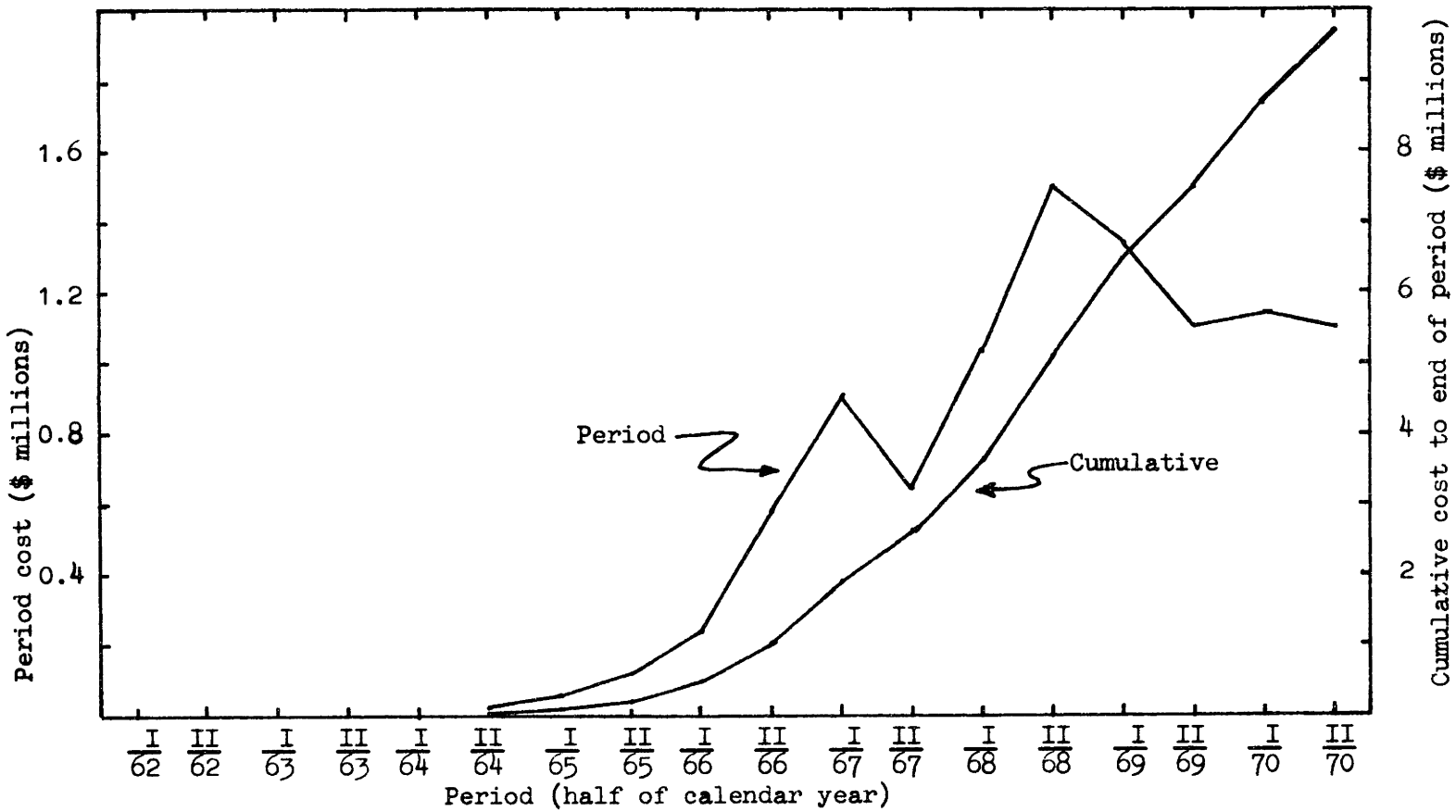

Figure 15. Period and cumulative cost of testing. (exclusive of computer cost)

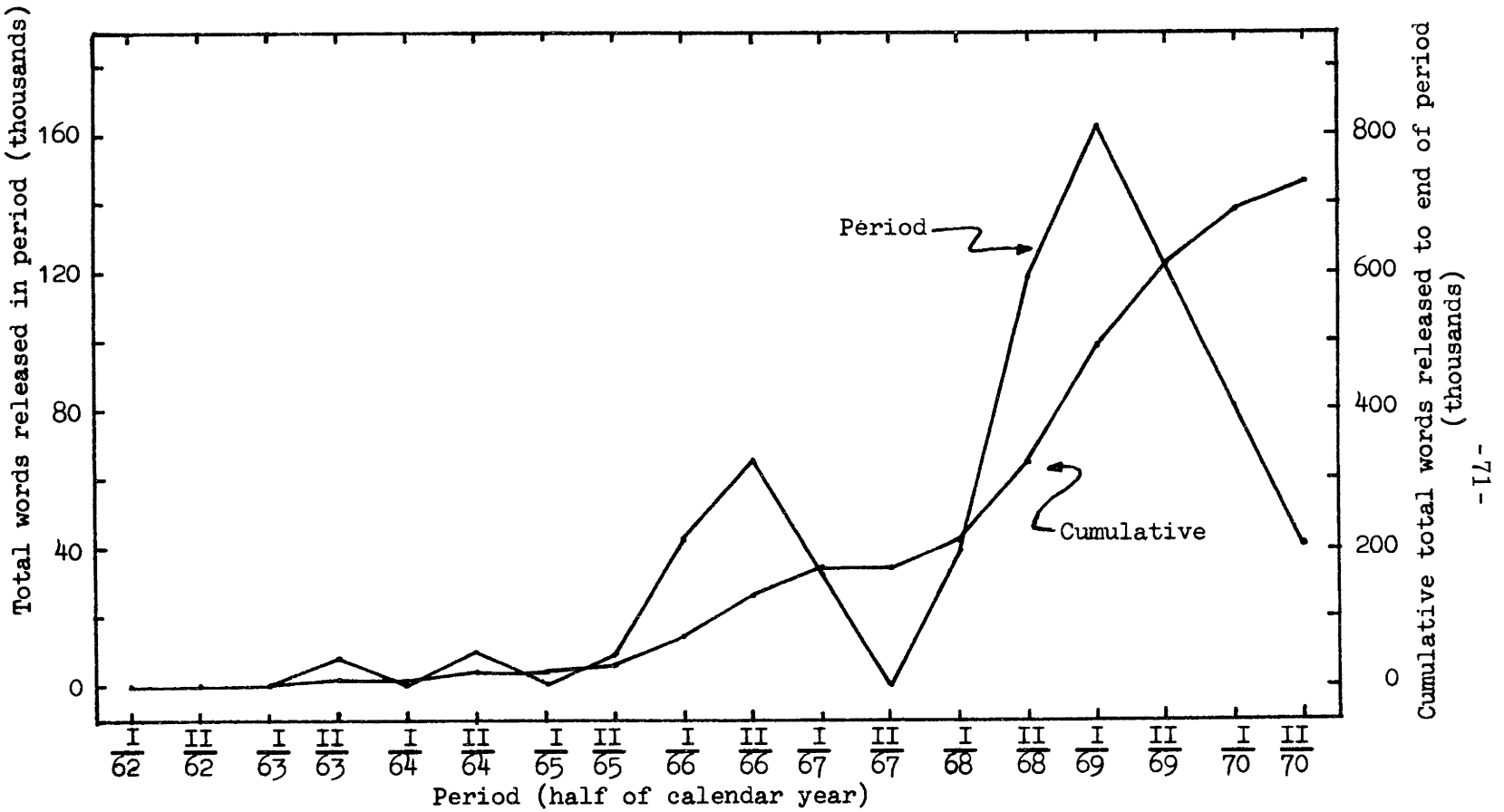

Figure 16. Total words released, by period and cumulatively.

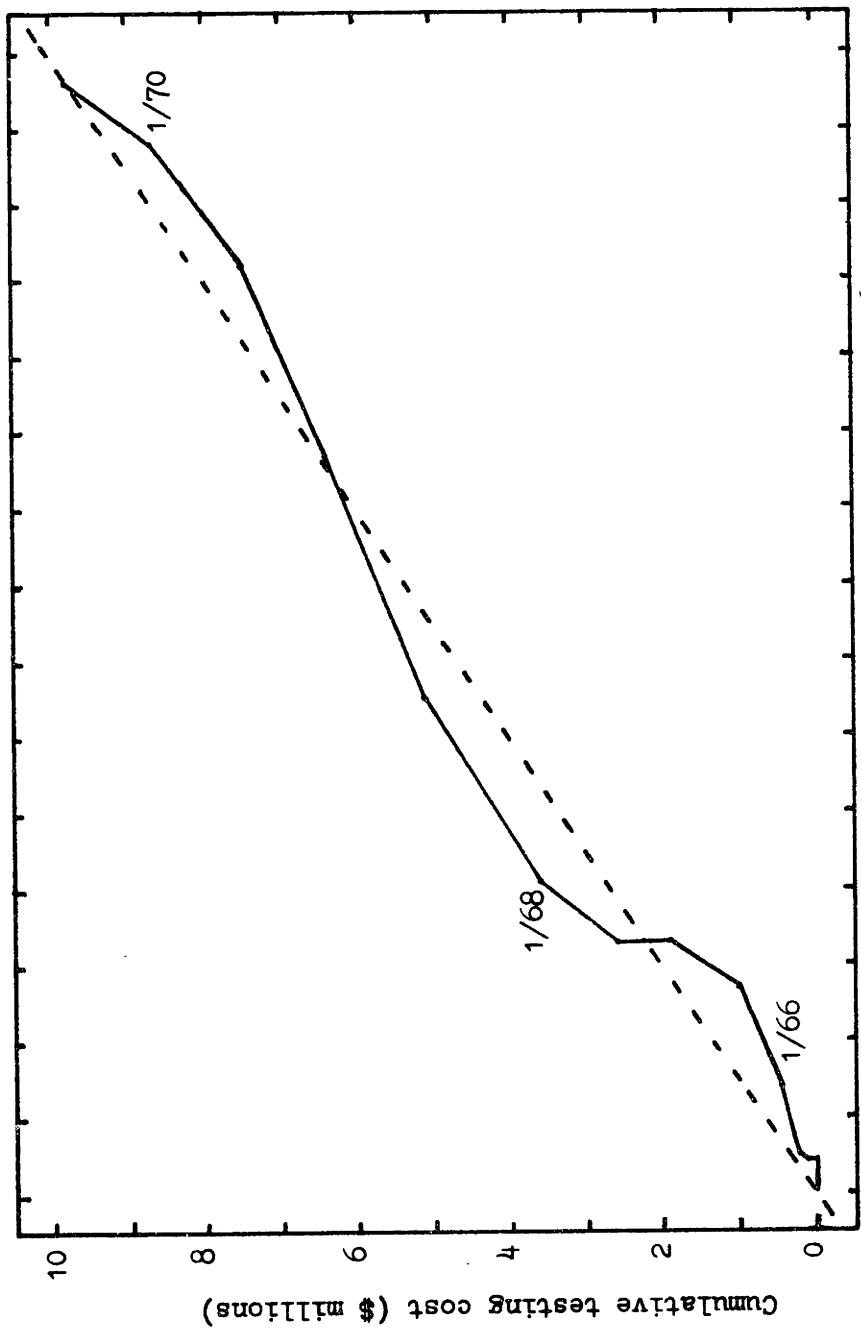

Figure 17. Cumulative testing cost vs cumulative total words.

It is interesting to note that even though the cost per word for testing is much lower than that for coding ($13.43 vs $48.92), the total cost of the testing function for Apollo software has been about the same as the cost of analysis and coding. This is shown in figure 3, page 23. Testing, exclusive of computer usage, was 14.2 percent of the total cost, while analysis and coding were 13.5 and 15.8 percent respectively.

5.3.4 Computer cost

Computing facilities are divided into two areas, digital and hybrid. These are two different installations operated by two different groups. The hybrid facility is a system which matches an AGC unit against a digital and analog simulation of the spacecraft and environment. It is an engineering test facility used extensively for software testing, mission verification, and crew training.

The digital facility is the "normal" computation center used by analysts and programmers. The equipment has changed over the years, but currently consists of an IBM 360/75 with extensive peripheral equipment. At the peak of the effort in 1968 and 1969 there were two such machines running around the clock to support software development.

The simplest fit of the cost of these two computer groups to the output measures has been found to be a simple multiple of the sum of the expenses for analysis, coding, and testing. The cost of digital and hybrid facilities have been summed and plotted in Figure 18. The sum of the costs for analysis, coding, and testing is shown in Fgiure 19. The two cumulative

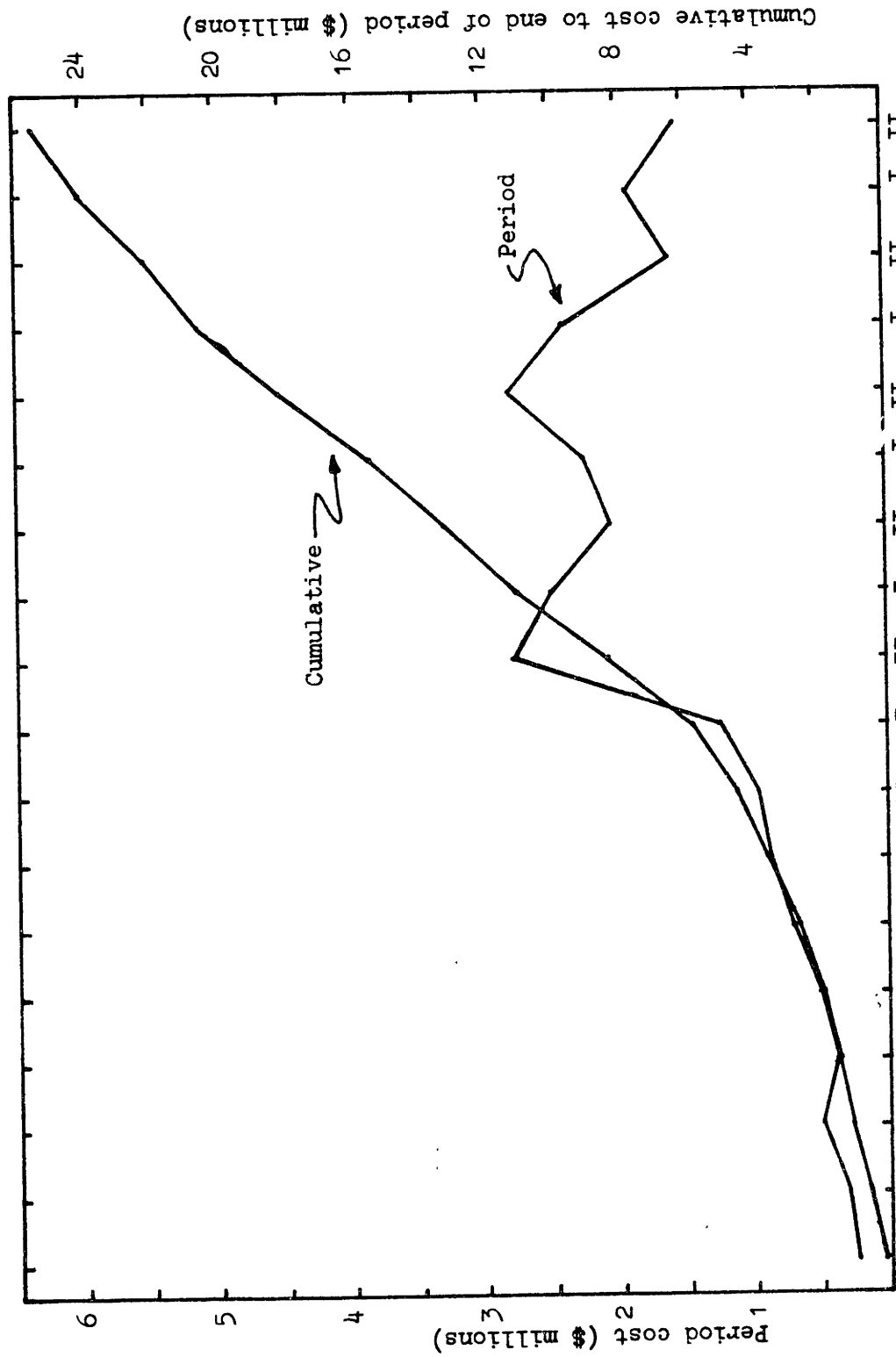

Figure 18. Cost of digital plus hybrid facilities by period and cumulatively.

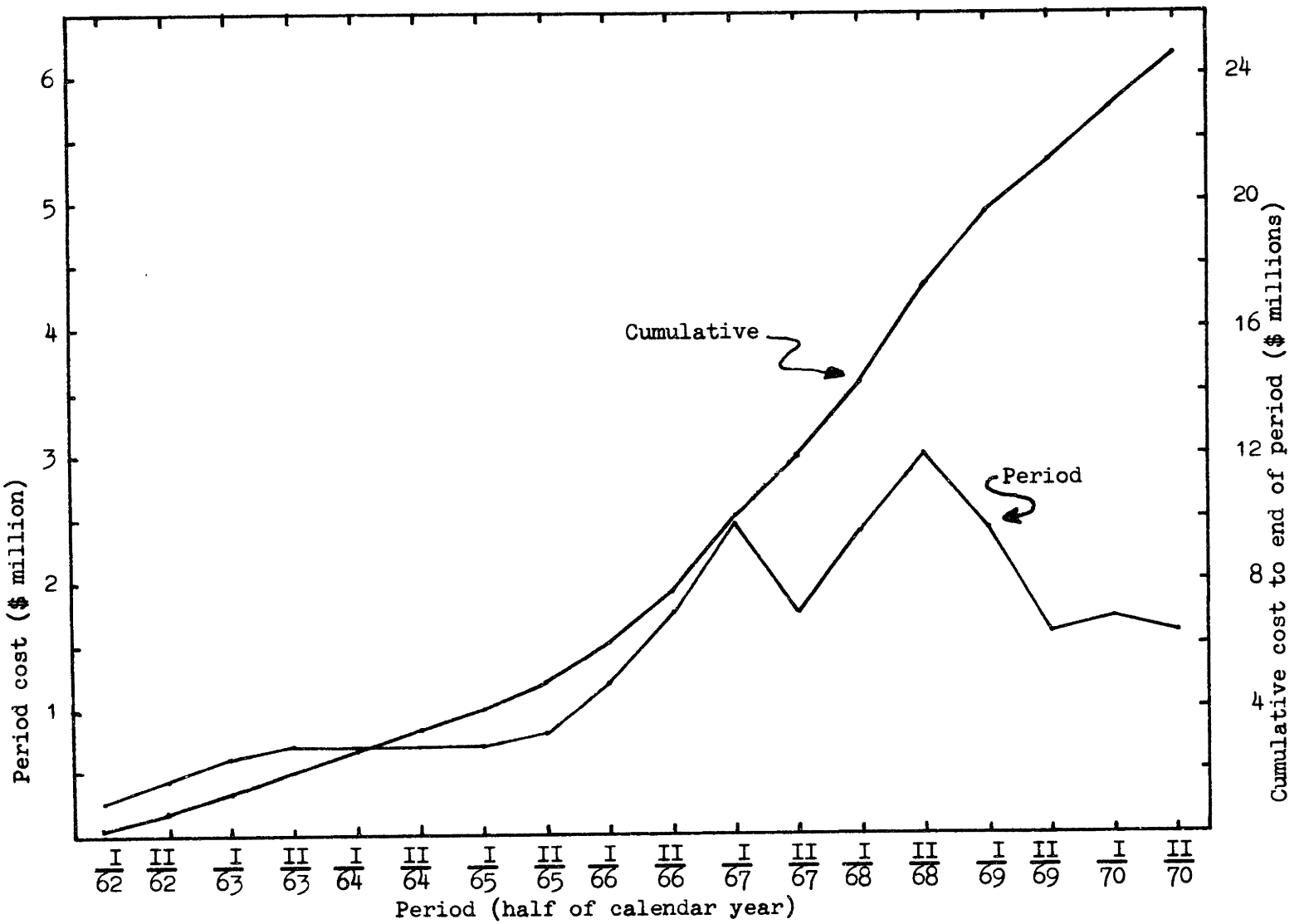

Figure 19. Period and cumulative cost of (analysis + coding + testing).

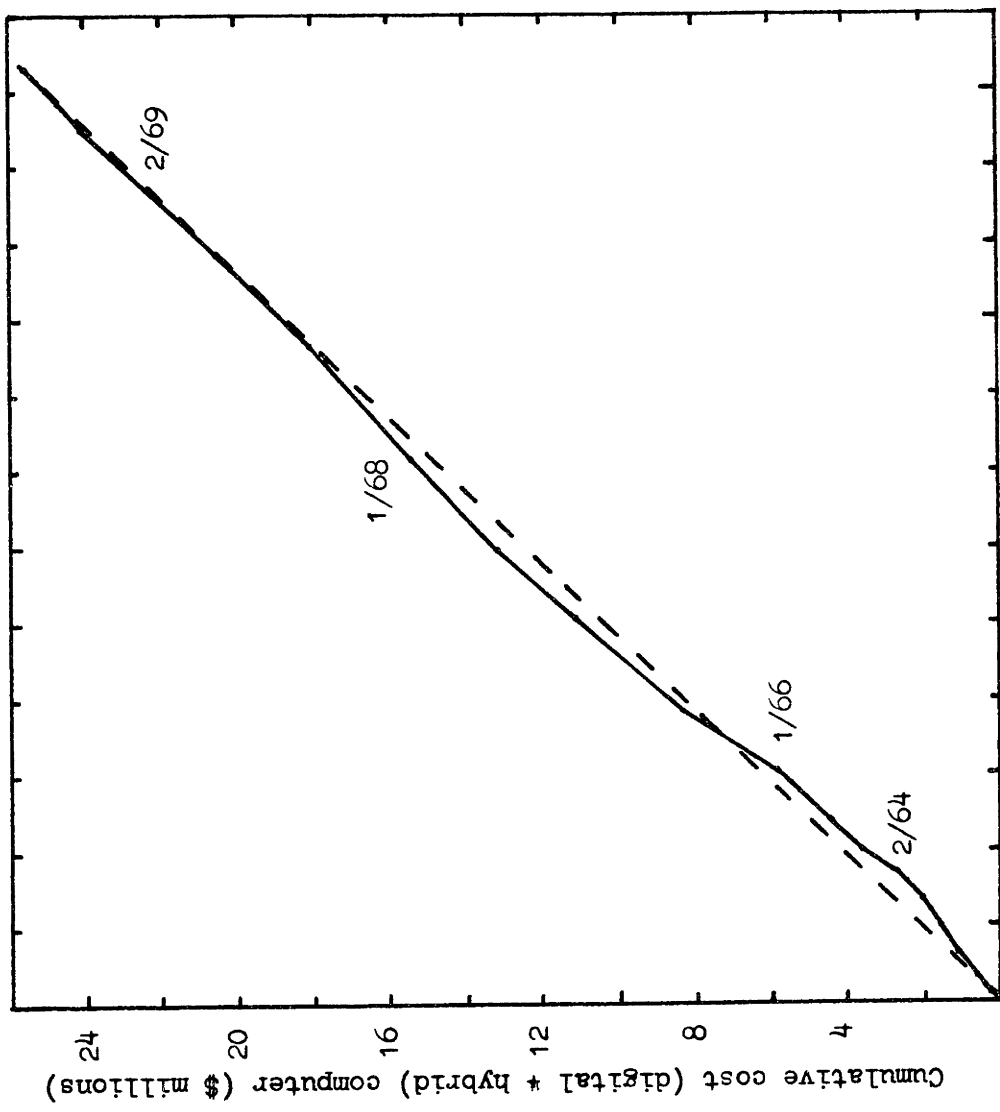

Figure 20. Cumulative computer cost vs cumulative (analysis + coding + testing).

cost curves are plotted against each other in Figure 20. Over the years, the cost of computers has averaged about 1.04 time the sum of analysis, coding, and testing.

This relationship asserts, in effect, that the average programmer working on AGC software used some average amount of computer time which was relatively constant over time. This is probably not quite correct. In particular, during the early years of the project, the computer group systems programmers (who are included under the heading of Computer cost) were very active on Apollo related work. They were developing the simulation models of the AGC, the spacecraft, the environment, and putting together the operating systems for the big machines. More recently such work has dropped to low maintenance levels. The ratio of system programming cost to machine time cost has probably gotten substantially lower as the project has progressed.

5.3.5 Documentation cost

The documentation cost for this effort is shown in Fgiure 21. The relationship D = \$0.17(A+C+T) has been used in the model.

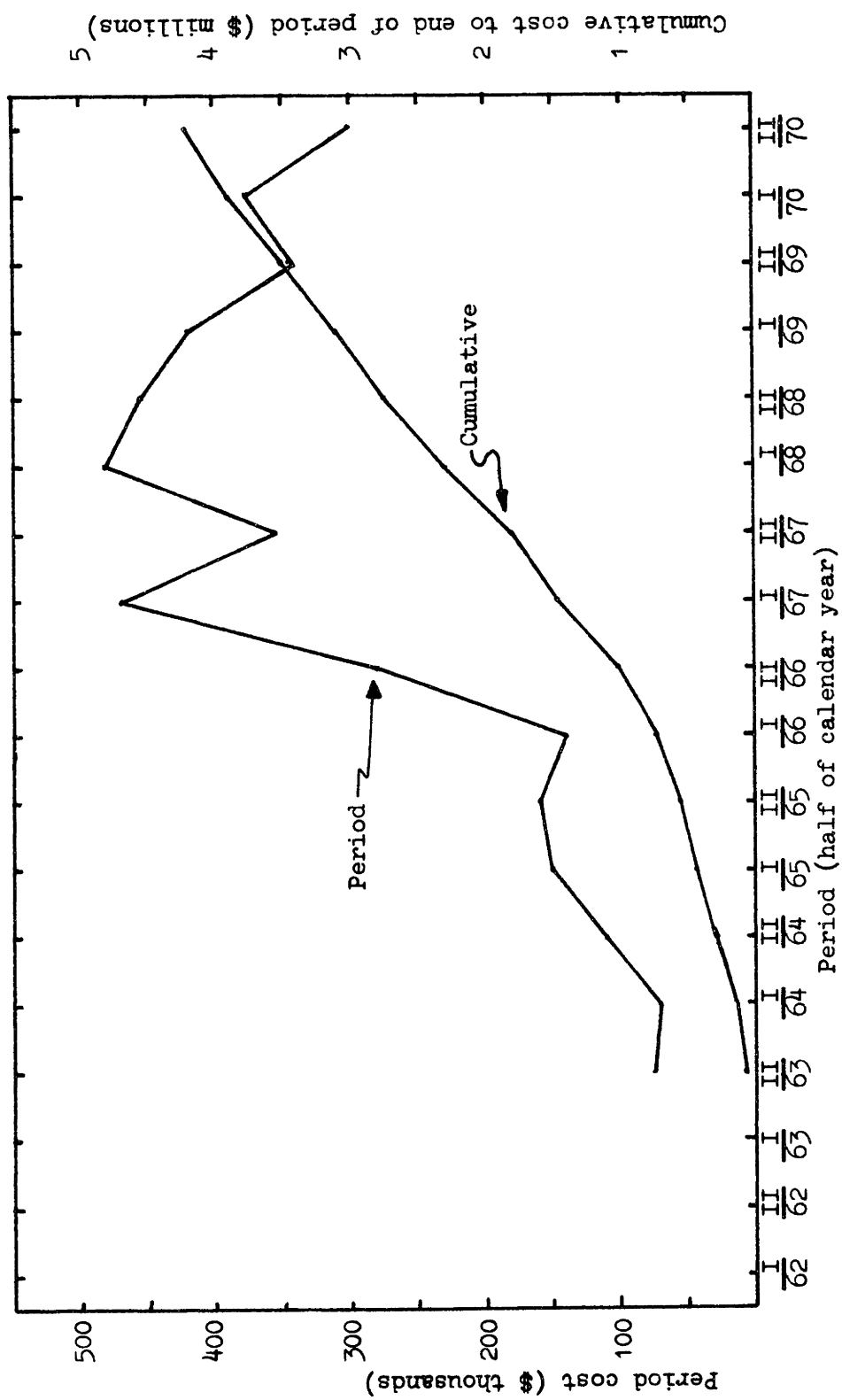

Figure 21. Period and cumulative cost of documentation.

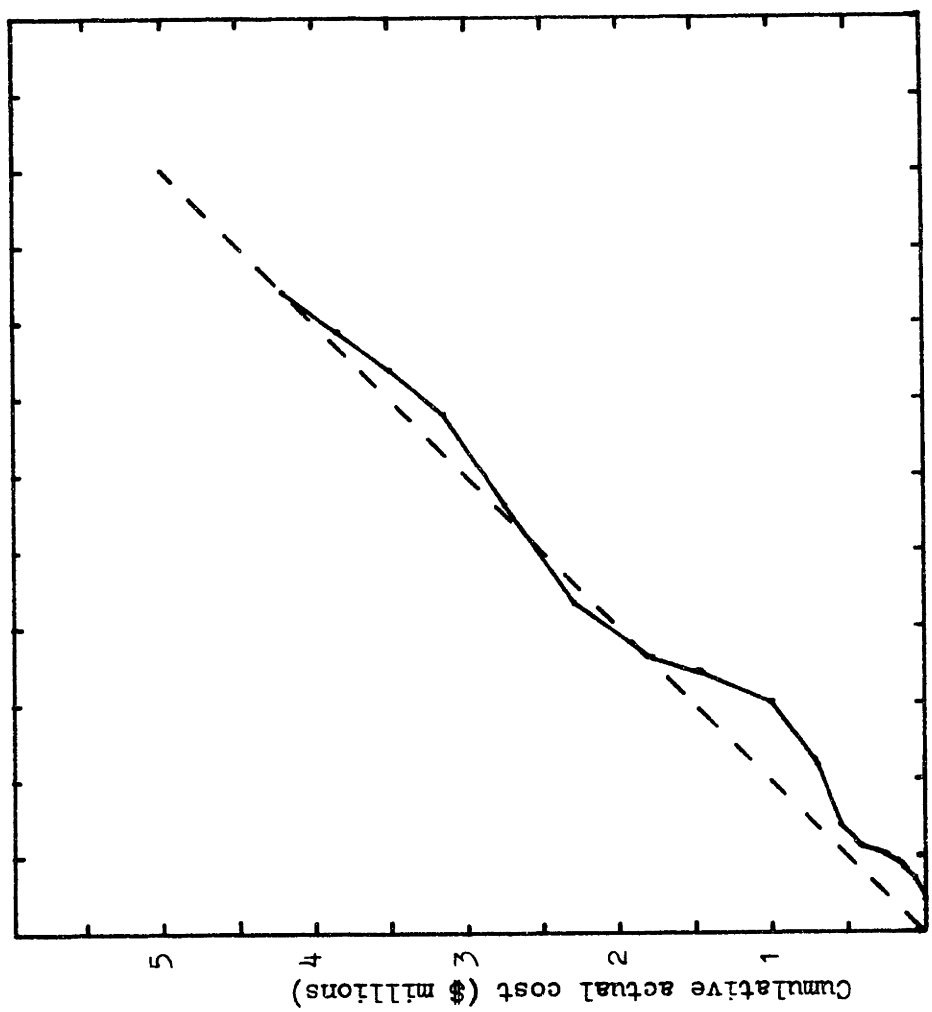

Figure 22. Actual vs model documentation cost.

The actual and model prediction documentation costs are compared in Fgiure 22. The documentation effort for this project has been quite extensive and has involved considerable preparation of specifications, an activity which might be classified differently on other projects.

5.3.6 Management cost

Management cost has been taken as proportional to (A+C+T), but is spread evenly over the life of the project. This is perhaps not quite realistic, but is a better fit to actual experience than modelling management cost as proportional to some of the other variables or factors over time. Management cost has averaged $0.11 per dollar A+C+T. Actual and model management costs are shown in Figure 23.

5.4 Smoothing

The raw cost for each period has been used as an entry into an exponential smoothing equation to obtain a smoothed cost for that period. The weighting factors have been set at 0.5 for the raw estimate and 0.5 for the smoothed estimate

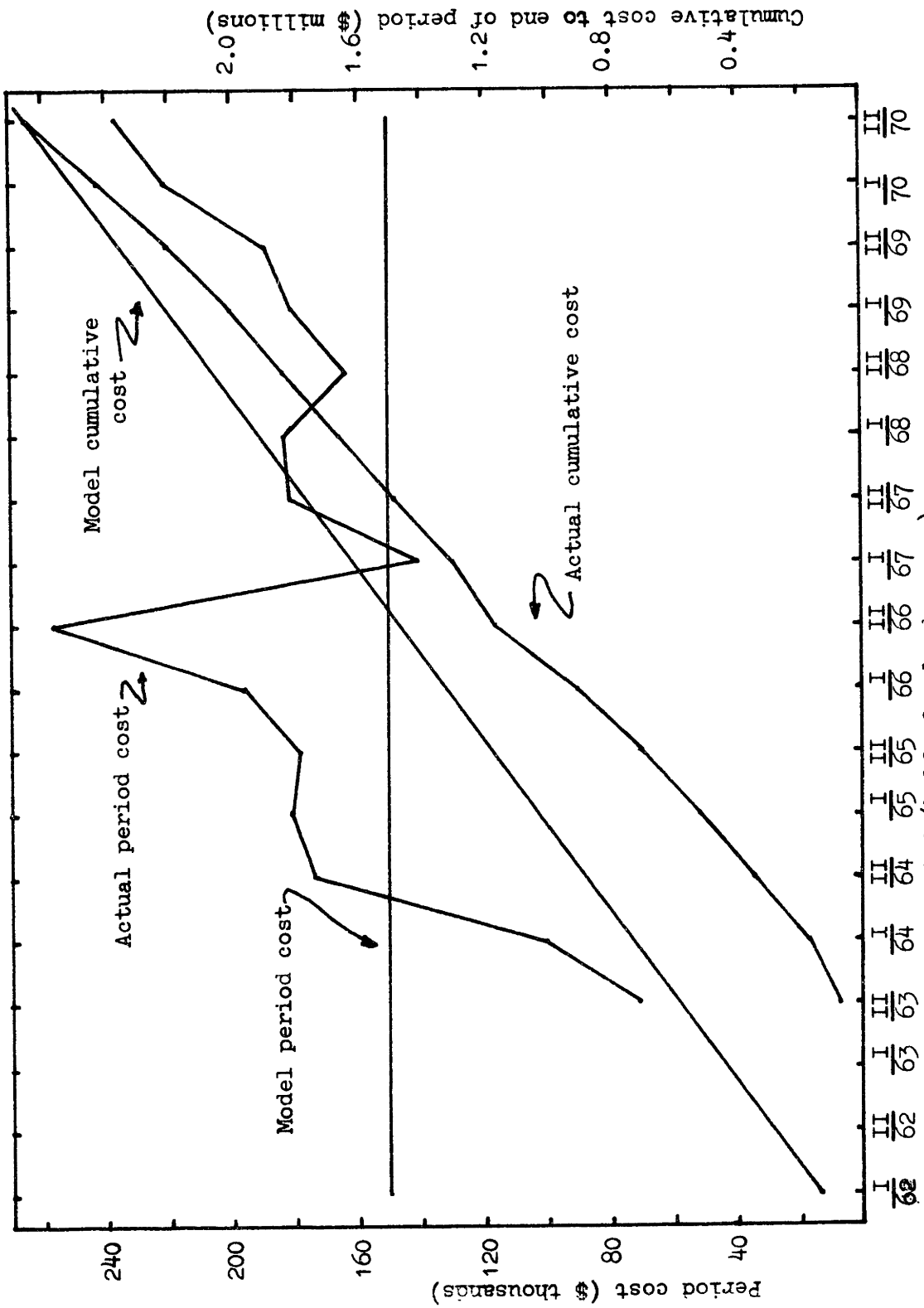

Figure 23. Actual and model management cost by period and cumulatively.

of the previous period.

Where Y_t is the raw estimated cost in period t, the smoothed estimated cost $\overline{Y}_t$ is :

$$\overline{Y}_t = 0.5\ \overline{Y}_{t-1} + 0.5\ Y_t$$

Smoothing is desirable for two reasons. First, it lessens the effect of the lumpy, discontinuous occurrence of output. The data for this model has been used as six month summaries in order to provide some averaging of output over time. For most releases, however, time and effort has been accumulated over a much longer period. Further, between release and the mission there are several months which are used for extensive high level testing.

A second benefit of smoothing is that it recognizes organizational inertia. It takes time to wind up or down expenses. The acquisition and layoff of computers, other equipment, and personnel are all subject to ead time and termination commitments. Considerations such as this were influential in the particular choice of weighting factors.

The smoothed costs are a closer fit to actual costs than are the raw predictions. Figures 1 and 2 (pages 19 and 20) illustrate this.

5.5 Results of the Model

The term by term predictions and the raw and smoothed total costs are compared with actual costs are shown in Figures 1, 2, and 9 through 23. Actual costs data is shown in Table 2 and in greater detail in the Appendix. Numerical data generated by the model are given in Table 3.

The biggest errors in the model are due to erratic timing of releases. In January, 1967, the tragic capsule fire killed three astronauts. This event delayed the Apollo project for approximately a year. There was only one release in the first half of 1967 (abbreviated I/67) and none in the second half (II/67) Thus the raw predicted expense (Figure 2) for 1967 falls much below actual expenditures. The smoothed cost prediction does considerably better, but cannot overcome the full effect of zero output at a time when work was in fact continuing.

A year later, in II/68, precisely the opposite effect occurs. Three large releases with many new words force the raw predicted cost about one third higher than the actual cost. The smoother cost again does considerably better. Smoothing, however, should not be expected to completely correct for such gross and unintentional changes in output.

TABLE 3 (Page 1 of 2). Term by Term Prediction of the Model Equation. (Dollars in Thousands)

PERIOD (HALF/YEAR)	1/62	2/62	1/63	2/63	1/64	2/64	1/65	2/65	1/66
Analysis Cost	365	366	365	366	365	365	366	365	366
New Words/Period	0	0	0	7978	0	2750	0	8500	24750
Coding Cost $48.92/New Word	0	0	0	390	0	135	0	416	1211
Total Words/Period	0	0	0	7978	0	10451	0	8500	42303
Testing Cost $13.43/Word	0	0	0	107	0	140	0	114	568
Computer Cost 1.04(A + C + T)	380	380	380	897	380	666	380	931	2230
Documentation Cost 0.17(A + C + T)	62	62	62	147	62	109	62	152	365
Management Cost	150	150	151	150	150	151	150	150	151
Cost for Period	958	958	958	2057	958	1565	958	2128	4889
Cumulative Cost	958	1916	2874	4931	5889	7454	8412	10540	15429
Smoothed Cost/Period	479	718	838	1448	1203	1384	1171	1650	3269

TABLE 3 (Page 2 of 2). Term by Term Prediction of the Model Equation. (Dollars in Thousands)

PERIOD (HALF/YEAR)	2/66	1/67	2/67	1/68	2/68	1/69	2/69	1/70	2/70
Analysis Cost	365	365	366	365	366	365	366	365	366
New Words/Period	22550	13200	0	23100	40480	15015	949	1842	7190
Coding Cost $48.92/New Word	1103	646	0	1130	1980	735	46	90	352
Total Words/ Period	65393	32488	0	38528	118179	161927	122050	81000	40250
Testing Cost $13.43/Word	878	436	0	517	1587	2175	1639	1088	541
Computer Cost 1.04(A + C + T)	2441	1505	380	2093	4090	3405	2133	1605	1308
Documentation Cost 0.17(A + C + T)	399	246	62	342	669	557	349	262	214
Management Cost	150	150	151	150	150	151	150	150	151
Cost for Period	5337	3349	958	4599	8842	7387	4683	3561	2930
Cumulative cost	20766	24115	25073	29672	38514	45901	50584	54145	57075
Smoothed cost/Period	4303	3826	2392	3495	6169	6778	5730	4646	3788

These two periods (a low estimate in II/67 followed by a high estimate in II/68) are the most dramatic instances of a more general phenomena. The predicted costs tend to run low for a few periods, then exceed actual.

It is obvious to an intelligent observer, as poposed to a dumb equation, what is happening. Output as counted in the data is lagging the input effort by more than the periodicity used here. The spreading of analysis and management costs and the exponential smoothing of the raw predicted cost both serve to mitigate this phenomena. In this particular case they do not, and can hardly be expected to be, completely solve the problem.

It is not really too serious a problem. If one looks at the cumulative expenses over the life of the project, these period to period errors tend to compensate each other. The tracking of the actual and modelled expenses is quite good in the long run. In any real situation, the human who interprets the model could recognize temporary distortions and make appropriate adjustments by hand to achieve a more realistic time/expenditure profile.

A more sophisticated approach would be the measurement of output continuously. In retrospect for this project that is not possible. For a project in the planning stage it should be possible to translate the delivery requirement into estimates of the production required in each period. For reporting purposes during a project this approach would require careful cooperation by the programmers through a well designed measurement system to an intelligent controller. Such use of the model might prove a valuable method of control and project monitoring.

5.6 Singular Characteristics of Apollo Software

Section 1.2 noted that Apollo software/computer programming differed considerably from "normal" software projects. The lack of objective standards for programming makes such comparisions dependent on the experience of the writer and reader. The observations herein are based on conversations with programmers and managers in the Draper Laboratory.

Computing system aboard aerospace vehicles face many constraints imposed by their service and environment. They must be small, light weight, and draw little electrical power. They must perform dependably in a rugged environment. They

often operate in a real-time, process control mode. The man-machine interfaces must be cleverly designed to provide good communication of real time data while using small, rugged displays that consume little computational overhead. Most importantly, such computers must be reliable from both a hardware and software point of view. Paradoxically, as the computational systems increase in complexity and responsibility, reliability becomes both more critical and harder to comprehensively verify.

The AGC was originally conceived as primarily an aid for off-line navigational calculations. Two effects converged to enlarge the role of the computer. As the complexity of the spacecraft and mission operations became apparent, it was necessary to transfer more duties to the computer. Secondly, designers came to appreciate the power and flexibility of a software programmed digital control system as contrasted with a hardwired analog system. Thus the expected amount of Guidance, Navigation, and Control duties increased from project conception to actual missions and the notion of the optimum degree of software based automation of these functions shifted toward greater computer responsiblity. The AGC was enlarged several

times, though without a basic change in its design, from the initial specification to the eventual 39K words of memory.

Despite this enlargement, the machine remained marginally adequate in terms of eraseable memory, fixed memory, and computational speed. The success of the Apollo missions makes obvious that such problems were overcome. It is equally clear to Laboratory personnel that working around these capacity limitations of the machine caused a substantial increase in the effort (which means cost) required. For example, the administrative overhead was increased by the need to precisely control memory usage. Also, the time required for comprehensive testing was substantially increased by the intricate complexity of coding created for minimum memory usage and minimum execution time. During the lunar landing, for example, the computer is running at about 90% of maximum computational capacity. As the programs become more complicated, the likelihood of unplanned effects of minor changes becomes greater.

5.7 The Effects of Time Pressure

Time has several significant effects on a project. If the time available to deadline is less than optimal but still feasible, then the project will cost more due to the diminishing marginal returns for added resources, perhaps compounded by lower quality of the additional resources.

At the other extreme, additional time available beyond the "minimum" required to complete the project will almost certainly be consumed in additional development and testing. This is not necessarily bad, given the dependence of the crew upon proper performance of the GN&C system. It is quite a natural feature of a project such as this in which a standing team produces a series of outputs as needed over several years. In the long run, the organization expands or contracts to meet needs. In the short run, cost is almost linearly proportional to time available to deadline.

To carry this argument a bit further, several people in the Laboratory feel that an excellent cost estimation technique is to estimate the size of the project team required, then multiply the cost of keeping and supporting that team by the time available to delivery of the product.

There is no way to make a quantitiative estimate of the effects of time pressure or computer limitations on the cost of Apollo computer programming. As a subjective estimate, there was great time pressure on the software project from 1965 until 1970. Since then, the pace of development work and Apollo launchings has slacked considerably. Computer limitations were a problem, but how this affected cost is difficult to judge, and in any case would be a strong function of the competence of the persons doing the work. The Laboratory is blessed with the finest people. Finally, keep in mind the very broad definition of software used in this study and the inclusion in the cost figures of all direct, indirect, and overhead costs.

CHAPTER VI

PREVIOUS WORK ON COST ESTIMATION

6.1 Introduction

There is a paucity of open literature bearing on the subject of this thesis. Most authors treat cost estimation only in passing if at all. While there are undoubtedly informal studies of historical cost data within companies, few of them have reached the open world except through casual conversation. There is a folklore of "cost per word" type estimates, but hardly a systematic science with documented data.

There is no substitute for experience in cost estimating. This author frankly lacks such experience. This makes it doubly hard to relate partially relevant literature to the model at hand.

The contribution of this paper is the case study and the style and coefficients of the model derived from the case study. With the possible exception of the SDC study reported below, I have seen no papers which use the general approach or the two count measures described in this thesis.

This chapter will give a brief description of the literature I have found on the subject of cost estimation.

6.2 The System Development Corporation Study

The System Development Corporation (SDC) conducted an extensive study of the cost factors in computer programming. They examined in considerable detail some 169 different programming efforts. The sample contained many small programs which were completed in a few months by a few programmers. The largest took 3 years, 2,700 computer hours, and 300 man months, but most were much more modest. Where possible, large development efforts were broken up into several discrete projects. The conclusions of the study are contained in Management Handbook for the Estimation of Computer Programming Costs by E. A. Nelson, published as a Technical Memorandum by SDC in 1967. Identically the same paper was republished in A Management Guide to Computer Programming by American Data Processing, Inc. in 1968.

According to Nelson, there are basically four methods of forecasting costs. To quote:

a: Specific Analogy, where costs for a new item are estimated by using the known costs for a similar item produced earlier.

b: Unit Price, where the cost of a new item is estimated by the produce of the number of units to be delivered in the new item (e.g., number of instructions) and previously determined cost per unit

c: Percent of Other Item, where the cost of a new item is estimated as a predetermined percent of the cost of another item, e.g., the cost of computer program design, code, and test might be a fixed fraction of total computer programming costs.

d: Parametric Equations, where the cost of the new item is estimated from an equation which is a function of various characteriestics of the requirement for the item resource expected to be used and working conditions.

(Nelson, 1967, p. 25)

Nelson then continues:

Each of these methods is comparative; each is dependent upon an applicable data base from past experience, each assumes that the past is prologue. To estimate a particular job, each method may be used alone, or in combination with the others. They also may be applied at various levels of aggregation; e.g., to estimate computer programming costs, the total computer program may be estimated as a whole, or broken up into components, such as steps in the programming process. All estimates of costs, no matter how subjective they may appear to be, are actually based on one or more of the above four methods.

In the sections of this Handbook, corresponding to six activities or steps in computer programming, we present data to enable the user to make estimates using the last three methods cited: unit price, percent of other cost, and parametric equations...

(Nelson, 1967, p. 25)

As indicated, he regards the system development as a six step process:

In this handbook, we have divided the programming process into distinct steps, or activities, for planning and estimating purpose. From the technical manager's viewpoint, there are two reasons for breaking up the programming process into steps. Different steps may represent fundamentally different kinds of jobs with consequently different cost implications, such as different types of personnel, tasks, and/or locations. Also, if the completion of a step can be a clearly defined and identifiable event with a definitive end product, these events constitute milestones useful for control.

The steps making up the computer programming process, or project cycle, are assumed to be:

a. Preliminary Planning and Cost Evaluation

b. Information System Analysis and Design

c. Computer Program Design, Code and Test (production)

d. Information System Integration Test

e. Information System Installation and Turnover

f. Computer Program Maintenance

(Nelson, 1967, p. 27)

The SDC approach was the gather data on resource consumption and on every conceivable factor which might have affected cost on the 169 program developments they studied. The measures of resource consumption are man months and hours of computer time. About 100 factors were considered. Some of these were simple parameters such as whether or not the computer room was open to programmers or the type of programming

language used. Others were variables such as the total number of object instructions delivered.

A mass of data was received, and processed in several ways. A linear programming formulation was used to try to develop parametric equations. Various cost factors, such as man months of effort to produce 100 pages of documentation or 1,000 lines of source code, were computed. Various subsets of the data were examined to try to establish systematic cost patterns as a function of such things as the type of programming language used.

Direct comparisons of Nelson's results with my model are difficult and subjective. He and the others at SDC attempted to break projects into the smallest, but still complete, units possible. My model regards a massive effort which consumed 9 years as one project. The Apollo effort was a case of an intense effort with much technical and mathematical problem analysis all focused onto a quite small computer. Nelson's sample includes few if any aerospace computers. The intangible nature of effort devoted to software and the different requirements of different projects makes it extremely difficult for him to get a uniform response from the projects he samples and for readers

to determine where they stand relative to his summaries. The cost of Apollo software includes all capital and administrative expenses while the people responding to the SDC questionaire may have looked at more narrow direct costs or changes in variable costs caused by undertaking small development efforts. Included in the costs charged to AGC programs are all the costs of providing systems programming and simulation models and programming on the Laboratory's computers. The testing and user training and documentation supplied by the Laboratory go much beyond normal.

It may be instructive to make some very crude calculations comparing the two models. This exercise should not be taken too seriously without better knowledge of the exact circumstances of particular situations. For the cost category "Computer Program Design, Code, and Test" the median of all SDC samples showed about 4 man months (σ= 10 man months) and 15 computer hours (σ= 45 hours) per 1,000 source instructions. If we assume that a man month costs $3,000 and a computer hour $600, then the cost of this activity is $21 per instruction. Since coding and testing generally represent 1/5 to 1/3 of the total development cost, then there is an indicated cost of about $60

to $100 per instruction. Obviously such an estimate must be regarded as indicative at best. Depending on the people, the project, and the methods of counting costs and instructions, one could get answers ranging from much cheaper to infinitely more expensive. This range of numbers does fit well, however, with the general range of undocumented intuitive guesses the author has heard from presumably knowledgeable people.

Using the model of this thesis for the release of new words, one obtains a cost per instruction of about $230. The indicated ratio of Apollo software being 2 to 5 times more expensive than "average" ground-based information system programming fits nicely with the author's judgement and more importantly the judgement of others who have had experience in Apollo and other aerospace projects.

6.3 Aerospace Cost Per Word Estimates

At various times, personnel of the Draper Laboratory have come across cost per word estimates for other aerospace computer programming projects. In general, the numbers they mention range from about $60 per word for projects with multiple

releases of almost standard but flight specific programs to over $1,000 per word for what are probably largely "new word" programs on small computers. For "normal" projects, if one can say that there is such a thing, these managers tend to expect a cost of $200 to $500 per instruction.

Written references for such estimates are hard to find, and even if available informally would probably be considered as the propriatary information of the source. Careful documentation, including even such elementary information as how words and costs were counted, seems to be non-existent in open literature and rare in the obscure world of aerospace contractor's reports. The significant point is that the costs reported here for Apollo software are in the range regarded as typical by informal community expectations for such projects.

6.4 Other Cost Estimating Techniques

A large number of authors have given papers, books, and manuals on the general topic of software development management. Many of these, at one point or another, address the problem of estimating the cost of the project. Unfortunately, most can do little better than a few platitudes to the effect that one should

plan carefully and do a good job of trying to estimate cost and resource usage throughout the life of the project. However, there is some serious work which the novice administrator would do well to read. The following sources are suggested.

One section of the Proceedings of the 23rd National Conference of the Association for Computing Machinery (held in 1968) entitled Managing the Economics of Computer Programming contains six short papers on manageing projects and cost estimation. Two of the authors were associated with the SDC and drew from the study mentioned earlier. The papers presented at the session were rewritten a bit and published as a worthwhile book edited by George Weinwurm in 1970. Pietrasanta (1970b) has an excellent paper in this volume on sequential estimation of each fragment of the project through the development cycle. The reader who can find the proceedings or (preferably) the book will have the most useful material available other than the SDC report. A recent publication by Robert Benjamin (1971) also gives a good picture of the system development cycle and a summary of previous work of that general nature.

The general approach to cost estimating advocated by these sources is quite simple. The project is divided into as

many discrete modules as possible, and these modules are then traced through the software development cycle. A typical cycle is that postulated by the SDC report and shown on page of this thesis. While there are slight variations, a six step cycle is generally used and changes are more a matter of style and particular author's experience than real differences. This approach is felt to be useful in that it forces the manager to enumerate everything which must be done, rather than being content with a one shot-in-the-dark global estimate for the project. The basic approach is to itemize every unit of work, the multiply by an appropriate cost per unit and sum. The problem with these sources is that they fail to give even typical ranges for the costs of unit factors. This important information the estimator must supply from his own or his organization's experience. The great merit of the SDC report as compared to all other generally available material in the field is the quantity of data. The variance is wide, the data samples are now 6 to 9 years old, the information must be interpreted in light of the particular situation of the estimator, and the final report is heavy with data and not very inviting to read. Nonetheless, that report is a starting point.

CHAPTER VII

CONCLUSIONS

This thesis has presented a relatively simple mathematical model which relates the cost and the output of software development for the Apollo Guidance Computer.

The model gives separate estimates for six categories of effort, namely Analysis, Coding, Testing, Computer, Documentation, and Management. One can use these factors if desired or simply use the sum. The cost is predicted on the basis of two output measures, total words or instructions released and new words which have been coded since the previous release of the programs.

For a variety of reasons, the particular project which is the basis of this case study and model is not typical of "normal" ground based information system development projects. However, it is reasonable to believe that the general form of the model may have some general validity. To the authors knowledge, the exact style is unique. The model predicts costs which vary by an order of magnitude and which

change from heavy analysis in early years to high volume testing in later years.

Whether or not the model is specifically useful for cost prediction, this case study should give the inexperienced manager a benchmark to which he can compare his project.

With some modifications and appropriate coefficients, this model should be quite useful as a device for monitoring progress on projects. It is a frequent experience that software projects get dangerously behind schedule long before management becomes aware of the problem. A similar and more subtle problem is a sudden and unanticipated demand for extra resources near the end of a project which is on schedule from a simple time line point of view. The enormous consumption of computer time in "system tests" during the final period before delivery is a particular example. Pietrasanta (1970a) in particular mentions this problem of unbalanced resource demand over time.

The main intention of this thesis has been to present the model, the data behind it, and a careful description of the circumstances of Apollo programming. It is a case study. The

author will be satisfied is readers find it an interesting study and quite pleased if someone finds it useful as background in thinking about the problem of costing computer programming.

Useful future work on the topic of this thesis can take several forms. First, there is a great need for more case studies of programming efforts. The costs, outputs, and circumstances of programming projects should be published. This will only happen if some organizations are willing to lift the vail of confidentiality which hides cost figures. This author suspects that embarassment about costs or even a refusal to recognize them is a major reason that so few case studies have been published.

Secondly, investigators must analyze existing work and the hoped-for flow of new studies. There should be some common factors through many projects which can be modelled to provide better predictive information. Hopefully the model presented in this paper will be of some usefulness. At the least, data should be compiled in a form which makes it accessible and meaningful to the everyday operating manager who finds himself in the position of having to estimate computer programming costs.

REFERENCES

Association for Computing Machinery. 1968. Proceedings of the 23rd National Conference. Princeton: Brandon/ Systems Press.

Benjamin, Robert I. 1971. Control of the Information System Development Cycle. New York: Wiley-Interscience.

Johnson, Madeline S. with Giller, Donald R. 1971. MIT's Role in Project Apollo, Volume V, The Software Effort. Final draft for Report R-700 of the C. S. Draper Laboratory of M. I. T. Cambridge.

Jones, Malcolm M. and McLean, Ephriam R. 1970. "Management Problems in Large-Scale Software Development Projects." Industrial Management Review, Vol. 11, no. 3.

(Mimno, Peter) 1971. Control of Software Development Cost. Technical Report of the C. S. Draper Laboratory of M.I.T. Cambridge.

Nelson, E. A. 1967. Management Handbook for the Estimation of Computer Programming Costs. Technical Report, Systems Development Corporation, Santa Monica. Also printed as a chapter in: A Management Guide to Computer Programming. Detroit: American Data Processing Inc., 1968.

Pietrasanta, Alfred M. 1970a. "Resource Analysis of Computer Program System Development." On the Management of Computer Programming. Edited by George F. Weinwurm. Princeton: Auerbach Publishers.

_____ 1970b. "Functional Estimating of Computer System Development." On the Management of Computer Programming. Edited by George F. Weinwurm. Princeton: Auerbach Publishers.

Weinwurm, George F., ed. 1970. On the Management of Computer Programming. Princeton: Auerbach Publishers.

APPENDIX

This appendix gives detailed data on the costs and functions of each organizational group for each time period. As is true throughout this thesis, a period is six months, the first or second half of a calendar year. The various columns may not sum perfectly because of rounding.

Section 4.1 of this thesis gives a detailed description of the duties of the more important organizational groups. The titles or descriptions below are given for convenience in using the appendix.

23	Main Apollo Group
23A	Space Guidance Analysis
23B	Mission Program Development
23C	Control and Flight Dynamics
23D	Man/Machine Systems
23D	M&S Materials and Supplies, mostly equipment for mockups and the Hybrid Computer system. Included in 23D unless specifically mentioned.
23H	Hybrid Computing Division
23M	Senior project managers, later counted as part of 23P

23P	Project Management
23R	Radar (early in project)
23S	System Engineering
23T	System Test Group
25	Hybrid Computing Division, before name change to 23H
26	Gyro Development (early in project)
33	Digital Computation
34	Analysis (early in project)

Costs in First Half, 1962. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23	50	239	Analysis	84	200
			Coding	16	39
25	100	79	Hybrid Comp.	100	79
33	100	180	Digital Comp.	100	180
34	100	23	Analysis	100	23
	Total	522			

Costs in Second Half, 1962. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23	34	403	Analysis	75	302
			Coding	25	101
25	100	43	Hybrid Comp.	100	43
33	100	278	Digital Comp.	100	278
34	100	22	Analysis	75	17
			Coding	25	5
	Total	746			

Costs in First Half, 1963. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23	34	597	Analysis	75	448
			Coding	25	149
25	100	37	Hybrid Comp.	100	37
33	100	447	Digital Comp.	100	447
34	100	15	Analysis	75	11
			Coding	25	4
	Total	1,097			

Costs in Second Half, 1963. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	610	Analysis	65	397
			Coding	25	152
			Documentation	10	61
23D	12.5	21	Hybrid Comp.	100	21
23P	34	71	Management	100	71
23S	66	64	Analysis	80	51
			Documentation	20	13
23T	25	61	Analysis	50	31
			Coding	50	31
25	100	17	Hybrid Comp.	100	17
33	100	360	Digital Comp.	100	360
34	100	15	Analysis	100	15
	Total	1,219			

Costs in First Half, 1964. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	540	Analysis	65	351
			Coding	25	135
			Documentation	10	54
23D	12.5	42	Hybrid Comp.	100	42
23P	34	100	Management	100	100
23R	50	22	Analysis	100	22
23S	66	87	Analysis	80	70
			Documentation	20	17
23T	25	64	Analysis	50	32
			Coding	50	32
25	100	16	Hybrid Comp.	100	16
33	100	461	Digital Comp.	100	461
34	100	15	Analysis	100	15
	Total	1,348			

Costs in Second Half, 1964. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	508	Analysis	50	254
			Coding	35	178
			Testing	5	25
			Documentation	10	51
23D	50	224	Hybrid Comp.	100	224
23M	34	53	Management	100	53
23P	34	121	Management	100	121
23R	50	51	Analysis	100	51
23S	100	145	Analysis	60	87
			Documentation	40	58
23T	25	67	Analysis	50	33
			Coding	50	34
25	100	46	Hybrid Comp.	100	46
33	100	449	Digital Comp.	100	449
34	100	7	Analysis	100	7
	Total	1.670			

Costs in First Half, 1965. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	594	Analysis	40	238
			Coding	40	238
			Testing	10	59
			Documentation	10	59
23D	33	253	Hybrid Comp.	100	253
23M	33	58	Management	100	58
23P	100	123	Management	100	123
23R	33	35	Analysis	100	35
23S	100	180	Analysis	50	90
			Documentation	50	90
23T	10	35	Analysis	40	14
			Coding	40	14
			Testing	20	7
25	100	153	Hybrid Comp.	100	153
33	100	451	Digital Comp.	100	451
34	100	7	Analysis	100	7
	Total	1,888			

Costs in Second Half, 1965. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	728	Analysis	30	218
			Coding	45	327
			Testing	15	109
			Documentation	10	73
23D	34	245	Hybrid Comp.	100	245
23M	34	49	Management	100	49
23P	34	128	Management	100	128
23R	12.5	11	Analysis	100	11
23S	100	169	Analysis	50	85
			Documentation	50	84
23T	10	49	Analysis	30	15
			Coding	40	20
			Testing	30	14
25	100	57	Hybrid Comp.	100	57
33	100	653	Digital Comp.	100	653
34	100	10	Analysis	100	10
	Total	2,100			

Costs in First Half, 1966. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	1,026	Analysis	30	306
			Coding	45	461
			Testing	20	205
			Documentation	5	51
23D	34	119	Hybrid Comp.	100	119
23P	50	196	Management	100	196
23R	12.5	11	Analysis	100	11
23S	100	183	Analysis	50	92
			Documentation	50	91
23T	20	109	Analysis	30	33
			Coding	40	44
			Testing	30	33
25	100	113	Hybrid Comp.	100	113
33	100	1,008	Digital Comp.	100	1,008
34	100	1	Analysis	100	1
	Total	2,767			

Costs in Second Half, 1966. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	839	Analysis	45	377
			Testing	50	419
			Documentation	5	42
23B	100	826	Coding	70	578
			Testing	15	124
			Documentation	15	124
23D	34	261	Hybrid Comp.	100	261
23P	50	258	Management	100	258
23R	12.5	2	Analysis	100	2
23S	100	228	Analysis	50	114
			Documentation	50	114
23T	20	131	Analysis	30	39
			Coding	40	52
			Testing	30	39
25	100	1,131	Hybrid Comp.	100	1,131
33	100	1,416	Digital Comp.	100	1.416
34	100	3	Analysis	100	3
	Total	5,094			

Costs in First Half, 1967. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	622	Analysis	45	280
			Testing	50	311
			Documentation	5	31
23B	100	1,733	Coding	55	953
			Testing	30	520
			Documentation	15	260
23D	66	374	Hybrid Comp.	100	374
23P	50	141	Management	100	141
23S	100	295	Analysis	40	118
			Documentation	60	177
23T	34	202	Analysis	30	61
			Coding	40	81
			Testing	30	61
25	100	470	Hybrid Comp.	100	470
33	100	1,648	Digital Comp.	100	1,648
	Total	5,485			

Costs in Second Half, 1967. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	479	Analysis	45	216
			Testing	50	239
			Documentation	5	
23B	100	1,212	Coding	55	667
			Testing	30	364
			Documentation	15	182
23D	66	373	Hybrid Comp.	100	373
23P	50	181	Management	100	181
23S	100	249	Analysis	40	100
			Documentation	60	149
23T	34	160	Analysis	30	48
			Coding	40	64
			Testing	30	48
25	100	171	Hybrid Comp.	100	171
33	100	1,506	Digital Comp.	100	1,506
34	100	5	Analysis	100	5
Total		4,336			

Costs in First Half of 1968. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	523	Analysis	40	209
			Testing	50	262
			Documentation	10	52
23B	100	1,490	Coding	55	819
			Testing	30	447
			Documentation	15	223
23C	100	457	Analysis	30	137
			Coding	30	137
			Testing	40	183
23D	66	313	Hybrid Comp.	90	282
			Documentation	10	31
23D M&S	66	127	Hybrid Comp.	100	127
23P	50	183	Management	100	183
23S	100	219	Analysis	20	44
			Documentation	80	175
23T	34	200	Analysis	10	20
			Coding	20	40
			Testing	70	140
25	100	407	Hybrid Comp.	100	407
33	100	1,434	Digital Comp.	100	1,434
34	100	2	Analysis	100	2
Total		5,355			

Costs in Second Half, 1968. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	491	Analysis	40	197
			Testing	50	246
			Documentation	10	49
23B	100	1,531	Coding	55	842
			Testing	35	536
			Documentation	10	153
23C	100	562	Analysis	30	169
			Coding	30	168
			Testing	40	225
23D	100	488	Testing	70	341
			Hybrid Comp.	20	98
			Documentation	10	49
23D M&S	100	73	Hybrid Comp.	100	73
23P	50	164	Management	100	164
23S	100	255	Analysis	20	51
			Documentation	80	204
23T	34	214	Analysis	10	21
			Coding	20	43
			Testing	70	150
23H & 25	100	164	Hybrid Comp.	100	164
33	100	2,446	Digital Comp.	100	2,446
Total		6,388			

Costs in First Half, 1969. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	406	Analysis	40	162
			Testing	50	203
			ocumentation	10	41
23B	100	1.236	Coding	50	618
			Testing	40	494
			Documentation	10	124
23C	100	365	Analysis	30	109
			Coding	30	109
			Testing	40	146
23D	100	447	Testing	70	313
			Hybrid Comp.	20	89
			Documentation	10	45
23D M&S	100	132	Hybrid Comp.	100	132
23P	50	181	Management	100	181
23S	100	263	Analysis	20	53
			Documentation	80	210
23T	34	203	Coding	10	20
			Testing	90	183
23H & 25	100	221	Hybrid Comp.	100	221
33	100	1.964	Digital Comp.	100	1,964
Total		5,418			

Cost in Second Half, 1969. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	324	Analysis	40	130
			Testing	50	162
			Documentation	10	32
23B	100	634	Coding	40	254
			Testing	50	317
			Documentation	10	63
23C	100	140	Analysis	30	42
			Coding	20	28
			Testing	40	56
			Documentation	10	14
23D	100	532	Testing	70	372
			Documentation	10	53
			Hybrid Comp.	20	106
23H	100	471	Hybrid Comp.	100	471
23P	50	189	Management	100	189
23S	100	220	Analysis	20	44
			Documentation	80	176
23T	34	207	Coding	10	21
			Testing	90	186
33	100	1,020	Digital Comp.	100	1.020
	Total	3,737			

Costs in First Half, 1970. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	229	Analysis	30	69
			Testing	60	138
			Documentation	10	23
23B	100	854	Coding	40	342
			Testing	50	427
			Documentation	10	85
23C	100	139	Analysis	20	28
			Coding	30	42
			Testing	40	56
			Documentation	10	14
23D	100	492	Testing	70	344
			Documentation	10	49
			Hybrid Comp.	20	98
23H	100	417	Hybrid Comp.	100	417
23P	50	221	Management	100	221
23S	100	253	Analysis	20	51
			Documentation	80	202
23T	34	193	Coding	10	19
			Testing	90	174
33	100	1,365	Digital Comp.	100	1,365
	Total	4,165			

Costs in Second Half, 1970. (Dollars in Thousands)

Group	Software As % of Groups Total Exp	Software Dollars	Function	Function As % of Software	Function Dollars
23A	100	125	Analysis	20	25
			Testing	60	75
			Documentation	20	25
23B	100	698	Coding	40	279
			Testing	50	349
			Documentation	10	70
23C	100	165	Analysis	20	33
			Coding	30	49
			Testing	40	66
			Documentation	10	17
23D	100	320	Testing	70	224
			Hybrid Comp.	20	64
			Documentation	10	32
23H	100	197	Hybrid Comp.	100	197
23P	50	237	Management	100	237
23S	100	198	Analysis	20	40
			Documentation	80	159
23T	34	436	Coding	10	44
			Testing	90	393
33	100	1,288	Digital Comp.	100	1,288
	Total	3,665			